PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Facultad de Ciencias e Ingeniería

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

EVALUACIÓN DE EQUIPOS 802.11 PARA INSTALACIONES RURALES DE LARGA DISTANCIA

Tesis para optar al título de:

INGENIERO DE TELECOMUNICACIÓN

Presentado por:

Josep Cursà Danés

Asesor:

David Chávez Muñoz

LIMA – PERÚ

Abril 2010

Resumen

Este proyecto se enmarca dentro del convenio y la buena relación institucional entre la Universidad Politécnica de Catalunya y la Pontificia Universidad Católica del Perú. Gracias a este convenio, a la colaboración de el Centre de Cooperació per al Desenvolupament (CCD) de la UPC, y al Grupo de Telecomunicaciones Rurales de la PUCP, este proyecto fin de carrera se ha podido llevar a cabo.

El trabajo consiste en hacer una serie de pruebas con diferentes combinaciones de software y hardware y ver su respuesta, con el objetivo de encontrar una combinación optima para comunicaciones inalámbricas de larga distancia en zonas rurales. Se centra principalmente en *802.11 para larga distancia* y en el uso de software y hardware libres. Se trabaja sobretodo con el sistema operativo *OpenWRT* y el controlador *madwifi* y se combina con las computadoras embebidas *Alix 2C0*, *Avila GW2348-4 Network Platform*, *Pronghorn SBC 250* y *Pronghorn Metro SBC*.

El proyecto se divide principalmente tres partes importantes. La primera parte consiste en analizar y estudiar las combinaciones de sistemas inalámbricos más óptimos, después se hacen diferentes pruebas para probar y analizar su funcionamiento y finalmente se obtienen conclusiones en función de los resultados.

Palabras clave: OpenWRT, Madwifi, computadoras embebidas de bajo costo y consumo, Alix 2C0, Avila GW2348-4 Network Platform, Pronghorn SBC 250, Pronghorn Metro SBC, IEEE 802.11, Wi-Fi, enlaces de larga distancia.

Per la meva família i en especial al pare i a la mare que fins i tot en els moments més difícils van creure en mi, i a tu Clara ja que sense tu potser tot això tampoc hagués estat possible.

Agradecimientos

Quiero mostrar mi agradecimiento a Gerson Araujo y a su familia por ser la mía durante todo éste tiempo, a todo el equipo del Grupo de Telecomunicaciones Rurales de la Pontificia Universidad Católica del Perú y en especial a River Quispe quien me ayudó desde el principio, a Reinaldo Baquerizo, Jose Luís Vargas y Rómulo Valencia.

También agradecer la colaboración del *Centre de Cooperacio per al Desenvolupament de la Universitat Politècnica de Catalunya (UPC)*, ya que sin su apoyo este proyecto no se hubiese podido llevar a cabo, y los consejos desde la distancia de Josep Maria Torrents Dolz, profesor de la UPC.

Gracias a todos.

Índice general

Índice de figuras

Capítulo 1: Introducción

Este trabajo, de fin de carrera, esta enmarcado dentro de los proyectos de investigación que lleva a cabo el *GTR (Grupo de Telecomunicaciones Rurales)* [1] de la *Pontificia Universidad Católica del Perú* [2]. El grupo de trabajo implementa sistemas de telecomunicaciones de larga distancia de bajo costo, para brindar servicios de telefonía y transferencia de datos a instituciones publicas ubicadas en zonas rurales, donde carecen estos servicios, de manera económica. Los sistemas pueden ser de bajo costo porque hasta ahora solo se exige tener al menos una velocidad de *2 Mbps* de extremo de extremo de la red para brindar estos servicios. El grupo de trabajo utiliza la tecnología Wi-Fi de larga distancia para implementar sus redes y ya tiene varias de ellas trabajando en la selva y sierra peruana. Estas redes están formadas por enlaces de larga distancia (de entre 10 km y 40 km). El equipo principal de los sistemas de telecomunicación es el enrutador Wi-Fi.

Actualmente el grupo de trabajo utiliza Wi-Fi de larga distancia, aun no utiliza WiMAX por ser una solución actualmente cara. El trabajo de fin de carrera se centra en evaluar distintas alternativas económicas de enrutador Wi-Fi de larga distancia para encontrar la mejor alternativa en rendimiento, economía, fiabilidad y robustez.

Evaluadas varias opciones, y partiendo de la calidad demandada por los enlaces y el costo de los mismos, se optó por un tipo de red inalámbrica formada mediante sistemas de telecomunicación OEM (Original Equipment Manufacturer). Con este tipo de sistemas se reduce el costo pero no se puede comparar con las opciones comerciales ya que éstas ofrecen mucha más velocidad de transmisión y ancho de banda. Sin embargo, se opta por sistemas OEM porque no se necesita mucha velocidad de transmisión. Se evaluarán distintos equipos con distintas tarjetas Wi-Fi, sistemas operativos y controladores para estas tarjetas.

Mediante este proyecto se espera contribuir en estrechar la brecha digital de las zonas rurales aplicando tecnología de bajo costo.

[1] http://gtr.telecom.pucp.edu.pe/

[2] http://www.pucp.edu.pe/

Capítulo 2: Análisis de la tecnología inalámbrica

2.1. Elección de la tecnología

El grupo de telecomunicaciones interconecta instituciones publicas utilizando enlaces inalámbricos Wi-Fi de larga distancia **[1]** brindando servicios de telefonía interna y acceso a la telefonía publica e Internet a un costo muy bajo comparado con soluciones satelitales. Se debe tener en cuenta que el GTR no es una empresa, es parte de un programa de apoyo para zonas rurales para ofrecer servicios de telecomunicaciones, por tanto al usar equipos de libre disposición cumple con las licencias de estos al no tratar como equipo comercial (especialmente del software).

El GTR implementa redes de datos en base a la tecnología Wi-Fi porque es una solución muy económica comparada a largo plazo con soluciones satelitales o cableadas. El equipo principal de un enlace Wi-Fi de larga distancia es el enrutador Wi-Fi **[2]**. El Grupo ha utilizado distintas soluciones de enrutadores Wi-Fi, siempre buscando soluciones libres y económicas pero con robustez, y ademas colaborando con la investigación tecnológica.

La tecnología inalámbrica elegida tiene que poder ofrecer la velocidad mínima requerida para soportar los distintos servicios de internet y de telefonía *IP*. Si se estima que una comunicación telefónica *IP* utilizando el *códec g726-32*, con tecnología inalámbrica *Wi-Fi* y con 100 tramas por paquete, requiere de unos *128 kbps*, y que la velocidad requerida por un equipo para la intercomunicación de datos internos sea de unos *256 kbps* (subida). Con estas cifras, en un enlace de 2 Mbps se dispondrá de alrededor de cinco comunicaciones telefónicas y cinco equipos accediendo a Internet con una calidad aceptable.

Siguiendo el trabajo del GTR se investigará a cerca de una tecnología de comunicación inalámbrica. Las dos principales tecnologías a ser usadas son *Wi-Fi en larga distancia* **[3]** y *WiMAX (Worldwide Interoperability for Microwave Access)* **[4]**.

Para la elección del mejor estándar a usar, a parte de las características técnicas de cada uno, también debemos tener en cuenta las necesidades de cada proyecto y los costos.

WiMAX está diseñado como una alternativa wireless al acceso de banda ancha DSL y cable, y una forma de conectar nodos *Wi-Fi* en una red de área metropolitana (MAN).

Puede dar cobertura a una área bastante extensa y esto lo hace adecuado para dar comunicación por ejemplo a ciudades enteras (70 kilómetros).

WiMAX, en distancias cortas (10 km), puede tener una velocidad de transmisión de hasta *70 Mbps* y también puede ser simétrico, lo cual significa que puede proporcionar un flujo de datos similar tanto de subida como de bajada (35 + 35 Mbps). El estándar trabaja a las frecuencias de 2,5 y 3,5 Ghz y también se utiliza una frecuencia libre de licencia a 5,4 Ghz (no estándar). El ancho de banda es de unos 20 MHz. En entornos reales, con una célula de seis kilómetros de radio, se han conseguido velocidades de hasta *20 Mbps*, y el ancho de banda se comparte por todos los usuarios de la célula.

Dentro de *WiMAX* se diferencian el estándar *"802.16d"* para terminales fijos, y el *"802.16e"* para estaciones en movimiento. Esto marca una distinción en la manera de usar el protocolo, aunque lo ideal es utilizar una combinación de ambos.

El llamado *WiMAX forum* es un grupo de empresas que se encargan de diseñar las normas y estándares de la tecnología *WiMAX* y de probar todos los nuevos componentes que van surgiendo. Actualmente lo forman más de 100 compañías.

Wi-Fi (*wireless 802.11*) como estándar esta limitado a unos 100 metros **[5]**. Por esta razón no se habla estrictamente de *Wi-Fi* sino de *Wi-Fi en larga distancia* porqué, para establecer un enlace de kilómetros, el estándar ha tenido que ser modificado. La principal ventaja del *Wi-Fi para larga distancia* es que puede establecer conexiones de unos 40 kilómetros y a un costo relativamente bajo, comparado con otras opciones como *WiMAX*.

La mayoría de enrutadores *Wi-Fi* estándares con las tres normas *802.11 (a, b y g)* es suficiente, pero para largo alcance se utilizan tecnologías especiales que obtienen el máximo rendimiento de la conexión. Un ejemplo puede ser el estándar *802.11-2007* que añade los modos 10 MHz y 5 MHz *OFDM* al estándar *802.11a* y extiende el tiempo de protección de prefijo cíclico de 0.8 microsegundos a 3.2 microsegundos, cuadruplicando la protección para la distorsión por trayectoria múltiple (multipath distortion).

Para complementar el estándar *Wi-Fi* de larga distancia los fabricantes aumentan la potencia de transmisión y se mejora la sensibilidad del receptor. Además se hace uso

de antenas de alta ganancia.

En algunos casos el protocolo *802.11* también puede ser modificado, con el riesgo de romper la interoperabilidad con otros dispositivos *Wi-Fi*.

Un punto a tener en cuenta para *Wi-Fi en larga distancia* es el acuso de recibo de los paquetes (*acknowledge*). Por defecto la distancia máxima entre el emisor y el receptor es de un kilómetro y medio más o menos, para mayores distancias el retardo forzará retransmisiones, así que para una comunicación optima se debe ajustar bien este parámetro en los equipos. El enlace más largo que se conoce de *Wi-Fi* sin amplificar fue en Italia y emisor y receptor estaban separados 304 kilómetros [1].

Los obstáculos son los mayores problemas al configurar un largo alcance en *Wi-Fi*. En campo los árboles, bosques y/o colinas degradan la señal de microondas y hacen que sea difícil establecer una línea de vista para la propagación. En ciudad los edificios bajan la conectividad y la velocidad de transmisión.

Así pues, *WiMAX* se ha diseñado para la cobertura de larga distancia y *Wi-Fi* para trasferencia de datos a corto alcance pero, con algunas modificaciones, esta ultima tecnología también se puede usar para comunicar puntos separados por larga distancia y con la ventaja que es más económica que una comunicación *WiMAX*.

WLAN 802.16 fue diseñado originalmente como una tecnología enfocada a los enlaces de larga distancia (hasta 50 Km) y células metropolitanas (7-10 Km), y usa el *DAMA-TDMA* corrigiendo ciertos problemas que presenta *WLAN 802.11* como el del nodo oculto por usar *CSMA/CA*. *WLAN 802.16* no es del todo buena opción porque esta tecnología presenta unos costos iniciales muy elevados (unos 2000-2500 USD por equipo en USA, MicroMAXe de Airspan), y no está muy difundido en el mercado peruano, es por ello que *WLAN 802.16* queda descartado pero no por falta de las buenas prestaciones que presenta.

Por lo tanto la tecnología apropiada dentro del contexto sería usar *WLAN 802.11*. Dentro de esta tecnología existen varias familias o estándares que incluyen distintas funcionalidades o modos de operación, las principales son las que se listan a continuación:

[1] http://www.adslfaqs.com.ar/wifi-record-de-304-kilometros-a-5-megas-de-velocidad/

- IEEE 802.11a: Banda 5.8 GHz con una velocidad de transmisión máxima de 54 Mbit/s.
- IEEE 802.11b: Banda 2.4 GHz con una velocidad de transmisión máxima de 11 Mbit/s.
- IEEE 802.11g: Banda 2.4 GHz siendo compatible con el IEEE 802.11b, con una velocidad de transmisión máxima de 54 Mbit/s.
- IEEE 802.11e: Incluye funcionalidades de QoS a nivel de la capa MAC.
- IEEE 802.11i: Especifica mecanismos de seguridad para redes inalámbricas.
- IEEE 802.11n: Introduce mejoras en el caudal efectivo mediante el uso de antenas MIMO (múltiple entrada, múltiple salida).

Entonces la tesis se centra en evaluar distintas soluciones Wi-Fi de largo alcance siempre teniendo en cuenta la mejor relación de costos, rendimiento y robustez.

2.2. Opciones del mercado wireless para larga distancia

En el mercado hay muchas opciones de equipos 802.11. Existen alternativas comerciales, alternativas mixtas, y alternativas totalmente abiertas (equipos OEM (Original Equipment Manufacturer)). A continuación se analizarán las alternativas que más se ajusten a las necesidades del proyecto y se elegirá una.

Trabajar con sistemas propietarios garantiza la ventaja que si surge algún fallo se pueda reparar de forma rápida, ya que es mantenido por una empresa. Como desventaja, no se pueden realizar configuraciones particulares ya que el sistema es propietario e impone restricciones. Como a característica a destacar, decir que las soluciones comerciales proporcionan un gran ancho de banda y mucha fiabilidad, respecto a los sistemas OEM (Original Equipment Manufacturer).

Los requisitos son que deben ser sistemas de bajo costo y con un rendimiento aceptable en larga distancia. De sistemas que se adecuen a estas necesidades hay muchos pero, como es de esperar, los que contrastadamente mejor responden a todo ello son los sistemas totalmente comerciales. En su contra aparece el costo. Un sistema totalmente comercial, dependiendo del sistema, pude salir dos o tres veces más caro que un sistema OEM (Original Equipment Manufacturer). Seguidamente se intentará ha-

cer una comparación técnica de las principales alternativas a escoger.

2.2.1. Soluciones comerciales

Se han encontrado tres opciones comerciales destacables que están dentro de los requerimientos planteados; *Smart Bridges*, *MikroTik* y *Lobometrics*. Por lo general los enrutadores Wi-Fi comprenden una computadora embebida, una tarjeta inalámbrica, el sistema operativo y su controlador correspondiente.

Los sistemas *Smart Bridges* y *Lobometrics* son sistemas totalmente compactos, donde hardware y software e incluso en algunos modelos la antena, se encuentran totalmente integrados y solo se permite configurar. Por contra, la solución *MikroTik* es un sistema semicompacto que si permite la modificación de su estructura. Cuando se adquiere una computadora embebida *MikroTik* esta ya viene con su software propietario, pero se pueden utilizar diferentes tipos de tarjetas inalámbricas 802.11 compatibles.

Solución Smart Bridges

El modelo de *access point sB3216* de *Smart Bridges* esta diseñado para trabajar en exterior y soporta estándares *IEEE 802.11a/d/h/i*, y además el modo de funcionamiento de estos estándares es mejorado con el modo de funcionamiento propietario del cual dispone el sistema. La firma no especifica que modificaciones plantea, respecto al estándar, para lograr la mejora. La siguiente imagen muestra el sistema embebido propuesto por *Smart Bridges*:

Figura 2.1: Solución comercial Smart Bridges sB3216.

Principales características:

- Procesador no indicado
- Sensibilidad máxima -108 dBm
- Potencia transmisión máxima 27 dBm
- Un interfaz radio en banda 5.8 GHz (4.800 MHz – 5.900 MHz)
- Dos puertos ethernet 10/100
- Antena integrada de ganancia 12 dBi
- Seguridad WEP, WPA y WPA2
- Consumo energético medio 9,6 W
- Rendimiento de 25 Mbps de tráfico TCP (enlace punto punto)
- Precio aproximado de un enlace 2400 USD

Las mejoras proporcionadas por esta opción permiten alcanzar enlaces punto a punto de hasta 40 kilómetros usando antenas de 31 dBi de ganancia en ambas radios. Se puede ver que el paquete embebido esta muy bien equipado y se adaptaría muy bien a las prestaciones que se necesitan pero su principal inconveniente es el precio.

La siguiente opción a tener en cuenta es la solución, también comercial, propuesta por *MikroTik*.

Solución MikroTik

A esta solución se le puede considerar una alternativa mixta entre equipos comerciales y equipos OEM. Esta opción pasa por el uso de la computadora embebida *RouterBOARD 333*, el sistema operativo propietario de *MikroTik*, *RouterOS* y una tarjeta inalámbrica que en este caso podría usarse la R52H (también de Mikrotik) o la Super Range 2 (de Ubiquiti Networks); en general cualquier otra tarjeta que sea de *chipset atheros*.

De entre las múltiples opciones de computadoras embebidas que ofrece *MikroTik*, se

elige la *RouterBOARD 333* básicamente por las siguientes características:

- Procesador PowerPC de 333 MHz
- Sensibilidad máxima -90 dBm
- Potencia transmisión máxima 25 dBm
- Puede trabajar en 2.4 GHz y 5.8 GHz
- Tres puertos ethernet 10/100
- Tres ranuras miniPCI (permite tres enlaces independientes)
- Software propietario RouterOS
- Rendimiento de unos 25 Mbps de tráfico TCP (enlace punto punto)
- Precio aproximado de un enlace (incluidas 2 tarjetas, pigtails y cajas) 800 USD

La siguiente imagen nos muestra la tarjeta embebida propuesta por *MikroTik*, la *RouterBOARD 333*:

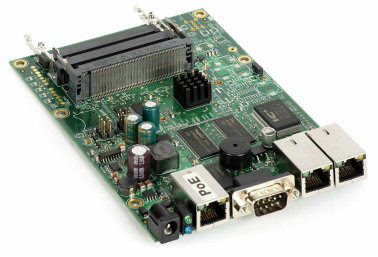


Figura 2.2: Solución comercial MikroTik RouterBOARD 333.

Lo interesante de la opción analizada es que se puede ajustar el modelo de tarjeta inalámbrica que más convenga a la necesidad y sobretodo también su costo, bastante más reducido que la solución propuesta por *Smart Bridges* y obteniendo rendimientos similares.

Solución Lobometrics

La solución de *Lobometrics* tiene como base el modelo *954HR*. La firma tiene mucha variedad para poder elegir pero se ha elegido esta por ser una de las que soporta tres interfaces cosa que permite poder tener tres enlaces independientes. Veamos las principales características:

- Procesador IBM RISC de 335 MHz
- Sensibilidad máxima -105 dBm
- Potencia transmisión máxima 26 dBm (por enlace)
- Permite tres enlaces independientes (integrado)
- Cuatro puertos ethernet 10/100
- Seguridad WEP, WPA y WPA2
- Consumo energético medio 10,92 W
- Rendimiento de 130 Mbps (enlace con línea de vista de 20 Km)
- Precio aproximado de un enlace (no incluye antenas) 3000 USD

La imagen siguiente es del sistema embebido propuesto por *Lobometrics*:

Figura 2.3: Solución comercial Lobometrics 954HR.

Esta solución, a pesar de ser cara, es una de las opciones más económicas que ofrece la marca y una de las que la relación calidad y prestaciones en frente del precio, esta mejor. Al Igual que la opción de *Smart Bridges*, esta empresa tampoco

especifica que modificaciones plantea, respecto al estándar, para lograr la mejora.

2.2.2. Solución OEM (Original Equipment Manufacturer)

Como ya se ha visto, los sistemas comerciales, a pesar de las buenas prestaciones ofertadas, tienen un inconveniente y es que son bastante costosos. La alternativa a estos sistemas es implementar un enrutador Wi-Fi por medio de equipos OEM. El mercado de los sistemas OEM es muy amplio y hay muchas opciones.

El precio de estas soluciones irá marcado por el costo de las antenas, de las tarjetas y sobretodo por la computadora embebida, pero la mayoría de combinaciones cuesta alrededor de los 300 USD.

Dos ejemplos de redes funcionando con equipos OEM son las redes [2] implementadas por el GTR en el río *Napo (Iquitos)* y la red *Willay (Cusco)*. La red *Napo* cuenta con 14 enlaces y ofrece una velocidad de transmisión de extremo a extremo de unos 2.5 Mbps, y por su lado la red *Willay* esta compuesta por 4 enlaces y ofrece una velocidad de extremo a extremo de 4.5 Mbps.

2.3. Sistemas OEM para larga distancia

Los sistemas implementados con sistemas OEM son los que, a nivel de costo, son más accesibles. El siguiente objetivo es encontrar una combinación de sistema que se adecue a la necesidad planteada.

Un enrutador Wi-Fi de larga distancia OEM se conforma por una computadora embebida, un sistema operativo, una tarjeta inalámbrica Wi-Fi de larga distancia, su controlador y una antena de alta ganancia. La estructura principal se puede ver en la siguiente imagen:

La principal característica que se necesita de la computadora embebida es que disponga de puertos *MiniPCI* y que ofrezca suficiente corriente a las tarjetas Wi-Fi de larga distancia (asegure alrededor de 1A por puerto).

[2] http://gtr.telecom.pucp.edu.pe/

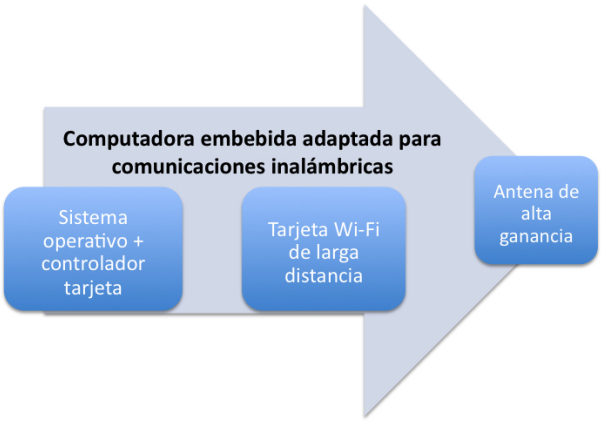


Figura 2.4: Estructura principal del sistema de telecomunicación.

En el mercado existen muchos sistemas operativos para computadoras embebidas, desde comerciales hasta de libre disponibilidad; la mayoría esta basada en Linux o BSD. No todos estos sistemas operativos son soportados por las computadoras embebidas que son ideales para implementar enrutadores Wi-Fi. Los sistemas comerciales no son muy caros, la mayoría está alrededor de los 50 USD por equipo instalado, pero las prestaciones son similares a los no comerciales. Quizá el comercial *Mikrotik RouterOS* sea una de las mejores opciones ya que ademas trae un controlador para tarjetas Wi-Fi de larga distancia y trabajan muy bien en sus propias computadoras embebidas (por ejemplo la RB433). Su rendimiento baja si se instala en otras computadoras embebidas y además aumenta la dificultad de instalarlo, ya que posee una licencia que permite sólo una instalación por equipo.

El controlador para las tarjetas Wi-Fi de larga distancia generalmente esta ligado al sistema operativo. Es muy probable que cualquier sistema basado en *Linux* (comercial o no comercial) permita instalar el controlador *madwifi* que se distribuye libremente. En los sistemas BSD este controlador esta poco mantenido y no se garantiza su uso.

Los equipos OEM de Wi-Fi de larga distancia elegidos para ser evaluados en este proyecto fin de carrera tienen buena relación de precio y robustez y rendimiento. Como sistema operativo, se buscarán los basados en *Linux* y que soporten el controlador *madwifi* por ser el de mejor rendimiento para larga distancia dentro de lo que es software de libre distribución. Mas adelante se verá el motivo de su elección.

2.4. Computadoras embebidas

Se buscaron distintas computadoras embebidas que estén adaptadas para enlaces Wi-Fi de larga distancia. La principal característica es que sus puertos *MiniPCI* ofrecen alrededor de 0.5A a 1.8A por puerto, que es la corriente que generalmente requiere una tarjeta Wi-Fi de larga distancia cuando usa la máxima potencia. Otra característica es que tenga como mínimo 2 puertos *MiniPCI* para poder tener dos enlaces. De entre muchas se eligió:

- Alix 2C0
- Avila GW2348-4 Network Platform
- Pronghorn SBC 250
- Pronghorn Metro SBC

Uno de los principales motivos por los cuales se ha escogido estas computadoras embebidas es porqué en todas ellas se puede instalar algún tipo de *linux* y por tanto instalar el controlador *madwifi*.

2.4.1. Alix 2C0

Características técnicas

- Fabricante: PC Engines
- AMD Geode LX700, 433 MHz Procesador (basada en x86-64)
- 128 MB DDR DRAM
- Compact Flash Socket
- Almacenamiento en Compact Flash
- DC jack o POE pasivo, min. 7V - max. 20V

- 2 Mini-PCI slots Tipo IIIA/IIIB, LPC bus
- 2 Puertos Ethernet (Via VT6105M 10/100)
- Puerto serie RS-232 (DB-9 connector macho)
- Temperatura de operación de -20ºC a +50ºC
- Medidas: 6”x 6”(15.24mm x 15.24mm)

Figura 2.5: Imagen de la computadora embebida Alix 2c0 de PC Engines.

2.4.2. Avila GW2348-4 Network Platform

Características técnicas

- Fabricante: Gateworks Corporation
- Intel® XScale® IXP425^{TM}, 533MHz Procesador
- 64 Mbytes SDRAM
- Compact Flash Socket
- Almacenamiento en 16 Mbytes de Flash (interna)

- DC jack o POE pasivo, min. 9V - max. 48V
- 18W de potencia para 4 Type III Mini-PCI slots
- 2 Puertos Ethernet 10/100 Base
- Puerto serie RS-232
- Temperatura de operación de -40^{o}C a +85^{o}C
- Medidas: 4”x 6”(10.16mm x 15.24mm)

Figura 2.6: Imagen de la computadora embebida Avila GW2348-4.

2.4.3. Pronghorn SBC 250 - Dual Radio Wi-Fi Router Board Based on the Intel® IXP425 Network Processor

Características técnicas

- Fabricante: ADI Engineering
- Intel® XScale® IXP425^{TM}, 533 MHz Procesador
- 64 Mbytes SDRAM
- Compact Flash Socket
- Almacenamiento en 16MB Intel®StrataFlash^{TM}(P30) (interna)
- DC jack 5V o POE pasivo en 802.3af Mode A and B, o injectores pasivos, o nominal 48V (rango 36-60V)
- 6.5 W para cada uno de los 2 Mini-PCI slots

- 2 Puertos Ethernet 10/100 Base
- Una vía RJ-12.
- Temperatura de operación de 0^{o}C a +70^{o}C
- Medidas: 4.72”x 6.40”(11.98mm x 16.25mm)

Figura 2.7: Imagen de la computadora embebida Pronghorn SBC 250.

2.4.4. Pronghorn Metro SBC - Next Generation Quad Radio Intel XScale® Wireless Router Board

Características técnicas

- Fabricante: ADI Engineering
- Intel® XScale® IXP425^{TM}, 533 MHz Procesador
- 64 Mbytes SDRAM
- Compact Flash Socket
- Almacenamiento en16MB Intel®StrataFlash^{TM}(P30) (interna)
- DC jack o POE pasivo; 802.3af Mode A and B, o injectores pasivos, o nominal 48V (rango 36-60V)
- 25W para 4 Mini-PCI slots a cualquier combinación de 3.3V o 5V.

- 2 Puertos Ethernet 10/100 Base
- Puerto serie RS-232 (DB-9 connector macho)
- Temperatura de operación de -40^{o}C a +80^{o}C
- Medidas: 6.365”x 5.748”(16.16mm x 14.59mm)

Figura 2.8: Imagen de la computadora embebida Pronghorn Metro SBC.

2.5. Sistema operativo para computadoras embebidas y controlador para tarjeta Wi-Fi de larga distancia

En el mercado existen muchos sistemas operativos para computadoras embebidas, desde comerciales hasta de libre disponibilidad **[6]**; la mayoría esta basada en Linux y BSD.

- **BSD:** FreeBSD, Monowall, STYX, NetBSD, OpenBSD
- **Linux:** IPCop firewall, IPFire firewall, LEAF, Meshlium, OpenWRT, Voyage Linuxy, Zeroshell
- **Comerciales:** Ikarus OS, DD-WRT, Embed-it, Mikrotik RouterOS

Para la computadora embebida *Alix 2C0* inicialmente se decide considerar la opción de poder utilizar *Monowall* o *FreeBSD*, pero al usar el controlador *madwifi* en *BSD* acarrea muchos problemas y no hay demasiado soporte, así que se descartan estas

opciones.

De los sistemas comerciales y sistemas libres basados en *Linux* se escogen los siguientes:

- **Sistemas libres:** OpenWRT y VoyageLINUX.
- **Sistemas comerciales:** DDWrt, Ikarus OS y Mikrotik RouterOS.

Con todo lo visto, y con ayuda de recomendaciones del propio fabricante, finalmente se decide que para la computadora embebida *Alix 2C0* **(ver fig. 2.5)** los sistemas operativos mas adecuados son *OpenWRT* [3] y *Router OS* [4].

Según el fabricante, para la computadora embebida *Avila GW2348-4*, el mejor sistema operativo (y el que la computadora embebida lleva ya por defecto instalado) es *OpenWRT*. El fabricante, aunque con menos prioridad, también recomienda como alternativa utilizar *DD-WRT* o *Ikarus OS*.

Para la computadora embebida *Pronghorn Metro SBC* el fabricante sugiere utilizar el *OpenWRT* y, si se quiere una alternativa, se podría usar *Snapgear* [5], otro sistema abierto basado en *Linux*. Como estas computadoras embebidas usan un procesador basado en *XScale* los sistemas operativos como por ejemplo *VoyageGTR* **[7]** quedan descartados, ya que estos sistemas son para procesadores basados en *x86*.

Del sistema operativo *DD-WRT* decir que la base es la misma que *OpenWRT*, y lo que principalmente cambia es la inclusión de una interfaz web. Actualmente también hay otras opciones comerciales parecidas a *DD-WRT*, como *XWrt* o *Luci*.

En conclusión se elige OpenWRT como sistema operativo para utilizar en estas computadoras embebidas; se podría utilizar el comercial DD-WRT, que posee una interfaz gráfica para su configuración rápida, pero no es muy critico no disponer de ello. En los equipos basados en Linux se puede usar múltiples controladores para distintas tarjetas Wi-Fi de larga distancia; actualmente existe el proyecto *Linux Wireless* [6] donde se desarrollan controladores, se mejoran los existentes y se crean otros nuevos para

[3] http://www.openwrt.org/
[4] http://www.mikrotik.com/software.html
[5] http://www.snapgear.org/
[6] http://wireless.kernel.org/

gran cantidad de modelos de tarjetas Wi-Fi y están totalmente a libre disposición. El controlador *madwifi* [7] es el mas conocido y el que posee mejor rendimiento para larga distancia. Este controlador solo trabaja en tarjetas Wi-Fi que tienen *chipset Atheros*. Así pues, para completar el enrutador Wi-Fi que se busca, se seleccionaron tarjetas Wi-Fi para larga distancia basadas en *chipset Atheros*.

OpenWRT es un sistema operativo de licencia libre definido como una distribución de *Linux* para dispositivos embebidos, como por ejemplo *routers*. En un principio este sistema operativo fue limitado al modelo *Linksys WRT54G*, pero debido a su rápida expansión se ha incluido soporte para otros fabricantes y dispositivos.

En lugar de crear un sistema operativo estático, con *OpenWRT* se ha hecho un sistema flexible, muy completo y que ofrece administración de paquetes. Estas opciones liberan al usuario de las configuraciones proporcionadas por los vendedores y permite que cada uno pueda personalizar *OpenWRT* para su dispositivo con los paquetes que necesite y para la aplicación que necesite. *OpenWRT* es el marco para construir una aplicación sin la necesidad de tener un *firmware* completo, cosa que para los usuarios se traduce en la posibilidad de modificar las características del *firmware* al gusto del consumidor.

El desarrollo de *OpenWRT* evolucionó inicialmente gracias a la licencia *GPL (GNU General Public License)* [8], que impulsaba a todos aquellos fabricantes que modificaban y mejoraban el código, a liberar éste y contribuir cada vez más al proyecto en general. Poco a poco el software ha ido creciendo y se encuentran características implementadas que no tienen muchos otros fabricantes de dispositivos comerciales para el sector no profesional, tales como *QoS*, *VPN* y otras características que dotan a *OpenWRT* de un dispositivo realmente potente y versátil. El hardware donde corre *OpenWRT* no sólo se puede usar como *router*, sino también como servidor de archivo, nodo *P2P*, servidor de webcams, firewall o puertas de acceso *VPN* entre otros.

Kamikaze y White Russian son las dos versiones existentes del *OpenWRT*. *White Russian* es la versión más antigua pero también la mas estable, *Kamikaze* todavía se esta desarrollando y no es tan estable.

[7] http://madwifi-project.org/

[8] http://www.gnu.org/licenses/gpl.html

2.6. Tarjetas

Una vez hecho el análisis de cada computadora embebida, y enumerados los sistemas que mas cumplen las necesidades que se requieren, se ve que el sistema que en general esta más recomendado y que mejor se acerca al perfil que se esta buscando es el *OpenWRT*.

Entre las principales ventajas que ofrece el uso de software libre están la libertad de uso y su distribución, soporte y compatibilidad a largo plazo, uso de formatos estándar, corrección más rápida y eficiente de fallos, independencia tecnológica y el fomento de la libre competencia, y además es de libre disponibilidad.

En el mercado de las tarjetas de larga distancia la mayoría utiliza el *chipset Atheros* y por tanto son soportados por el controlador *madwifi*.

Estas tarjetas se han escogido principalmente por recomendación de los fabricantes y porqué tienen *chipset atheros*, además de ser especiales para larga distancia y también porqué son de alta potencia:

- Ubiquiti Networks Super Range 2 (SR2) [9]
- Mikrotik R52H [10]
- Ubiquiti Networks Xtreme Range 2 (XR2) [11]
- Engenius EMP-8602 + S [12]

A continuación se analizarán una por una las tarjetas listadas anteriormente y se enumerarán sus principales características:

2.6.1. Super Range 2 (Ubiquiti Networks)

SR2

[9] http://www.ubnt.com/products/sr2.php
[10] http://www.routerboard.com/pricelist/download_file.php?file_id=97
[11] http://www.ubnt.com/downloads/xr2_datasheet.pdf
[12] http://www.gowifi.co.nz/specs/EMP-8602%20Plus-S.pdf

- Chipset: Atheros AR5213, 4th Generation
- Radio Operation: IEEE 802.11b/g, 2.4GHz
- Interface: 32-bit mini-PCI Type IIIA
- Operation Voltage: 3.3VDC
- Antenna Ports: u.fl (main), MMCX (secondary)
- Temperature Range: -40C to +80C
- Security: 802.11i, AES-CCM & TKIP Encryption, 802.1x, 64/128/152bit WEP
- Data Rates: 6Mbps, 9Mbps, 12Mbps, 24Mbps, 36Mbps, 48Mbps, 54Mbps
- TX Channel Width Support: 5MHz / 10MHz / 20MHz / 40MHz
- RoHS Compliance: YES
- Avg. TX Power: 26dBm, +/-1dB
- Max Current Consumption: 1.10A, +/-100mA
- Indoor Range (Antenna Dependent): over 200m
- Outdoor Range (Antenna Dependent): over 50km
- Operating System Support: Linux (madwifi), WindowsXP, Windows2000
- Advanced Mobility / Quick Handoff: WindowsXP/2000 Utility with Enhanced Mobility Driver from Ubiquiti
- Cisco Support: CCX 4.0 Supported Driver/Utility also available from Ubiquiti

2.6.2. R52H (Mikrotik)

R52H

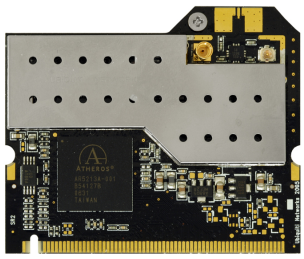


Figura 2.9: Imagen de la tarjeta Super Range 2 de Ubiquiti Networks.

- Chipset: Atheros AR5414
- Standards: IEEE802.11a, IEEE802.11b, IEEE802.11g
- Media Access: CSMA/CA with ACK architecture 32-bit MAC
- Security: Hardware-based 64/128 bit WEP, TKIP and AES-CCM encryption, WPA, WPA2, 802.1x
- Modulation:
  - 802.11b+g: DSSS, OFDM for data rate 30Mbps
  - 802.11a: OFDM
- Host Interface: Mini-PCI form factor; Mini-PCI Version 1.0 type 3B suggested only for motherboards that are produced after 2004
- Connectors: Two U.fl connectors
- Wi-Fi: WECA Compliant
- Certifications: FCC, EC
- Powering: 3.3V +/- 10 % DC; 800mA max (600mA typ.)
- Frequencies:
  - 802.11b/g: 2.192 – 2.507 (5 MHz step); 2.224 – 2.539 (5MHz step)

- 802.11a: 4.920 – 6.100 (5 MHz step)

- Transfer Data Rate:

  - 802.11b: 11,5.5,2,1 Mbps, auto-fallback
  - 802.11g (Normal mode): 54,48,36,24,18,12,9,6 Mbps, auto-fallback
  - 802.11g (Turbo mode): 108,96,72,48,36,24,18,12 Mbps, auto-fallback
  - 802.11a (Normal mode): 54,48,36,24,18,12,9,6 Mbps, auto-fallback
  - 802.11a (Turbo mode): 108,96,72,48,36,24,18,12 Mbps, auto-fallback

Figura 2.10: Imagen de la tarjeta R52H de Mikrotik.

2.6.3. Xtrem Range 2 (Ubiquiti Networks)

Esta es una de las dos tarjetas de alta potencia que se probaran, junto a la *Engenius EMP-8602 + S* que se analizará mas adelante.

XR2

- Chipset: Atheros AR5414, 6th Generation
- Radio Operation: IEEE 802.11b/g, 2.4GHz
- Interface: 32-bit mini-PCI Type IIIA
- Operation Voltage: 3.3VDC

- Antenna Ports: Single MMCX
- Temperature Range: -45C to +90C (extended temp. version up to +95C)
- Security: WPA, WPA2, AES-CCM & TKIP Encryption, 802.1x, 64/128/152bit WEP
- Data Rates: 6Mbps, 9Mbps, 12Mbps, 24Mbps, 36Mbps, 48Mbps. 54Mbps
- TX Channel Width Support: 5MHz / 10MHz / 20MHz / 40MHz
- RoHS Compliance: YES

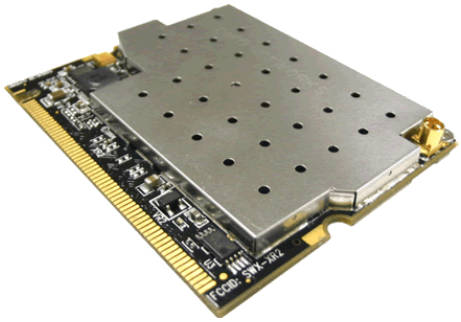


Figura 2.11: Imagen de la tarjeta Xtreme Range 2 de Ubiquiti Networks.

2.6.4. Engenius EMP-8602 + S

Por último analizar la segunda tarjeta de alta potencia que se usará. Decir que, tanto ésta como la anterior tarjeta, pueden llegar a una potencia de salida de *600 mW*.

Engenius EMP-8602 + S

- Up to 54Mbps data transfer rate
- Flexible designs for embedded systems or OEM project
- Fully interoperable with IEEE802.11a/b/g compliant products
- Updated 64/128-bit WEP engine for improved throughput
- Uses latest IEEE802.1x Client Support (EAP-TLS, EAP-TTLS)
- Advanced power management for extended battery life

- Improved indoor multipath distortion for higher link quality in an indoor environment
- Output Power up to 28dBm in 11b/g mode, 22dBm in 11a mode
- Extended tuning ranges for worldwide use and DFS/TPC for international operation
- Technical Specifications
    - Media Access Protocol – CSMA/CA
    - Regulation Certifications – FCC Part 15/UL
    - Modulation Technology – OFDM
    - 802.11: a/g OFDM (64-QAM, 16-QAM, QPSK, BPSK)
    - 802.11b: DSSS (DBPSK, DQPSK, CCK)
    - Security 64/128-bit WEP data encryption
    - WPA/WPA2- Wi-Fi Protected Access(64, 128-bit WEP with TKIP/AES, Pre-Share Key)
    - IEEE802.1x Client Support(EAP-TLS, EAP-TTLS supplicant)
- Physical Specifications
    - Form Factor – Mini-PCI Type IIIA
    - Dimensions – 59.60mm X 44.45mm
    - Antenna Connector – 2x U.FL Connectors
    - Operating Voltage – 3.3V +/- 0.15V

Con las tarjetas analizadas y las computadoras embebidas anteriores, y junto con el sistema *OpenWRT*, se probarán varias combinaciones de sistemas para ver su respuesta y sacar conclusiones al respecto. Se debe tener cuidado con las combi-

Figura 2.12: Imagen de la tarjeta Engenius EMP-8602 + S.

naciones de tarjetas con las computadoras embebidas ya que algunas de estas no soportan las tarjetas de alta potencia como la *XR2* **(ver fig. 2.11)** o la *Engenius* **(ver fig. 2.12)**.

Con estas combinaciones de software se harán distintas pruebas, tanto en laboratorio como en campo. Para las pruebas fuera del laboratorio, tanto en el campus universitario como en campo, se van a necesitar antenas. En el siguiente punto se analizan varias opciones de antenas a usar.

2.7. Antenas

Las antenas que se van a usar son antenas de alta ganancia especiales para distancias largas. La lista siguiente es de las principales antenas candidatas a ser usadas:

1. **Hyperlink RE11DS-NM**
    - Antena panel
    - Frecuencia: 2400-2500 MHz
    - Ganancia: 11 dBi
    - Ancho de haz horizontal: 60 grados
    - Ancho de haz vertical: 30 grados
    - VSWR: 1.5:1 avg.
2. **Hyperlink RE11DP-NM**

- Antena panel
- Frecuencia: 2400-2500 MHz
- Ganancia: 11 dBi
- Ancho de haz horizontal: 60 grados
- Ancho de haz vertical: 30 grados
- VSWR: 1.5:1 avg.

3. **Hyperlink HG2458-08LP-NF**
    - Antena panel
    - Frecuencia: 2300-6500 MHz
    - Ganancia: 8 dBi
    - Ancho de haz horizontal: 80 grados
    - Ancho de haz vertical: 60 grados
    - VSWR: 1.5:1 avg.
4. **Hyperlink HG2458-09P**
    - Antena panel
    - Frecuencia: 2400-2500 MHz / 5125-5850 MHz
    - Ganancia:
        - 9 dBi @ 2400-2500 MHz
        - 6 dBi @ 5125-5850 MHz
    - Ancho de haz horizontal: 65 grados
    - Ancho de haz vertical: 25 grados
    - VSWR: 1.5:1 avg.
5. **Hyperlink HG2458-14P-090**

- Antena sectorial
- Frecuencia: 2400-2500 MHz / 4900-5900 MHz
- Ganancia: 14 dBi
- Ancho de haz horizontal: 90 grados
- Ancho de haz vertical:
  - 2400-2500 MHz: 16 grados
  - 4900-5900 MHz: 8 grados
- VSWR: 1.5:1 avg.

6. **Hyperlink HG915G-NF**
   - Antena grilla
   - Frecuencia: 870 - 960 MHz
   - Ganancia: 15 dBi
   - Ancho de haz horizontal: 30 grados
   - Ancho de haz vertical: 19 grados
   - VSWR: 1.5:1 avg.
7. **Pacific Wireless GD9-18**
   - Antena grilla
   - Frecuencia: 900 - 928 MHz
   - Ganancia: 18 dBi
   - Ancho de haz horizontal: 16.5 grados
   - Ancho de haz vertical: 16.5 grados
   - VSWR: 1.5:1 avg.

Si se quiere conocer más sobre las características de estas antenas se puede ver los *datasheets* adjuntos en el anexo **A.3**.

A modo de conclusión del capítulo decir que, se probarán varias combinaciones del sistema operativo *OpenWRT* con el controlador *madwifi*, con las cuatro computadoras embebidas y las cuatro tarjetas vistas, y se harán pruebas en laboratorio y en campo. Para las diferentes pruebas también se escogerán las antenas adecuadas.

Capítulo 3: Ensamblado de los sistemas

3.1. Instalación de OpenWRT en las computadoras embebidas

Se puede decir que existen dos formas de instalar *OpenWRT*, dependiendo de si se instala en una memoria *Compact Flash* (caso de la *Alix 2C0*) o en un almacenamiento *Flash* interno (caso de las computadoras embebidas *Avila GW2348-4*, *Pronghorn SBC 250* y *Pronghorn Metro SBC*). Las computadoras embebidas *Avila* y *Pronghorn* ya llevan una *bios* por defecto, el *Redboot* [1]. *Redboot* es un *firmware* que se maneja con sus propios comandos.

Si se usa una memoria *Compact Flash*, se utiliza un adaptador de *Compact Flash* con conexión *USB*. En caso de que se use la memoria *Flash* interna de las computadoras embebidas, se hace mediante *TFTP*.

3.2. Kernels OpenWRT y transferencia a las computadoras embebidas

Cada computadora embebida usa un *kernel* de *OpenWRT* adecuado según su arquitectura y aplicaciones que se necesitan, entre otras cosas. En esta sección se hará referencia a las especificaciones de cada *kernel*, la manera como se compila, y como se transfiere a la computadora embebida **[8]**.

En este caso, para compilar los *firmwares* y para gestionar las computadoras embebidas remotamente, se ha hecho mediante una computadora con *Ubuntu 8.10* instalado.

Se empezará con la instalación del *OpenWRT* en la computadora embebida *Alix 2C0* **(ver fig. 2.5)**. Este equipo, a diferencia de los demás, es de arquitectura *x86*. Las otras tres computadoras embebidas son basadas en arquitectura *XScale* y eso implica que se debe generar otro tipo de *kernel* para ellas.

La computadora embebida Alix 2C0 **(ver fig. 2.5)** no cuenta con memoria interna así que, el *firmware* que se genere, se deberá transferir a una memoria *Compact Flash* externa. En las computadoras embebidas *Pronghorn* y la *Avila* se ubicará el

[1] http://sourceware.org/redboot/

kernel en su memoria interna.

Los pasos que se siguen a continuación son en *Linux* en el PC y son comunes para todas las computadoras embebidas.

Primero se debe instalar en *Ubuntu* los siguientes paquetes, que serán necesarios para la compilación del *kernel* del OpenWRT:

```
$ sudo apt-get install build-essential
$ sudo apt-get install gcc
$ sudo apt-get install git-core
$ sudo apt-get install minicom
$ sudo apt-get install lib64ncurses5-dev
$ sudo apt-get install lib64z1-dev
$ sudo apt-get install gawk
$ sudo apt-get install flex
$ sudo apt-get install subversion
```

Ahora se debe descargar el código fuente de *OpenWRT*:

```
$ git clone git://nbd.name/openwrt.git
```

Se crea una carpeta con el nombre *patches* y dentro un archivo llamado *defaults* con el siguiente código:

```
$ vim /patches/defaults
```

```
#!/bin/sh
uci batch <<-EOF
       # Configure the eth1 NIC from the ALIX.2C2
       # to act as a WAN port and get the IP address via DHCP
        set network.wan=interface
        set network.wan.proto=dhcp
        set network.wan.ifname=eth1
        commit network
EOF
```

Se modifica el archivo *feeds.conf* que se encuentra en la carpeta *openwrt*, (si el archivo no existe se debe crear). Como en lugar de usar *svn*, que está por defecto, se usará *git* se debe de añadir las dos siguientes lineas al archivo:

```
src-git packages git://nbd.ds10.mine.nu/packages.git
src-git luci git://nbd.ds10.mine.nu/luci.git
```

Se sigue ejecutando las siguientes ordenes:

```
$ mkdir -p files/etc/uci-defaults
$ cp -fpR patches/defaults files/etc/uci-defaults/
$ chmod a+x files/etc/uci-defaults/defaults

$ ./scripts/feeds update packages luci
$ ./scripts/feeds install -a -p luci
$ make menuconfig
```

Al ejecutar la última orden, *make menuconfig*, aparece un menú donde se pueden seleccionar los bloques del *kernel* que se quieren. En este menú es donde se especifica las características de la computadora embebida, los módulos que se quieren cargar en nuestro *kernel*, etc. Destacar el controlador *madwifi*, que es común para todas las compilaciones, y que permitirá al *OpenWRT* reconocer las tarjetas inalámbricas que se quieren usar.

3.2.1. OpenWRT para la computadora embebida Alix 2C0 (x86)

De las siguientes características remarcar que el sistema, que es *x86*:

```
Target System: x86 [2.6]
Target Profile: PCEngines Alix
Target Images
          jffs2: N
          ext2: N
Network
          hostapd: M
```

```
          wpa-supplicant: M
Kernel modules
          Filesystems
                    kmod-fs-ext3: M
          Network Devices
                    kmod-via-rhine: *
                    TODOS LOS DEMÁS...: N
          USB Support
                    kmod-usb-core: M
                    kmod-usb-ohci: M
                    kmod-usb-storage: M
                    kmod-usb2: M
          Wireless Drivers
                    kmod-madwifi: *
Administration
          LuCI Components
                    luci-admin-full: M
                    luci-app-ddns: M
                    luci-app-firewall: M
                    luci-app-ntpc: M
                    luci-app-qos: M
                    luci-app-samba: M
          LuCI Themes
                    luci-theme-openwrtlight: M
Utilities
          disc
                    cfdisk: M
                    swap-utils: M
          e2fsprogs: M
```

Una vez salvadas las características que queremos tener en nuestro sistema operativo, solo queda crearlo. Para ello se ejecuta el siguiente comando:

```
$ make world
```

Ahora solo queda esperar a que el PC compile el *kernel*. Según las características de la máquina que se use, esta puede tardar más o menos.

Una vez terminado el trabajo del PC y obtenidos los ficheros del sistema operativo se deben transferir a la *Compact Flash*. Se debe disponer de una lectora de memorias *Compact Flash USB* para establecer conexión con el PC.

Se conecta la memoria a la lectora y ésta al PC. Después se abre un terminal de *linux*, se identifica como *root* y se desmonta la unidad */dev/sdX* [2].

Con la memoria formateada se debe transferir el *kernel* que se encuentra en */alix/bin*. Primero se borra la partición de la *Compact Flash* y después se transfiere el sistema operativo. El archivo que se debe pasar se llama '*openwrt-x86-squashfs.image*'.

```
$ dd if=openwrt-x86-squashfs.image of=/dev/sda
```

Una vez ejecutados estos comandos ya se tiene el sistema operativo en la memoria. Solo queda conectarla a la computadora embebida y alimentarla para que arranque. Para acceder vía cable serie a la computadora embebida *Alix*, se hará mediante *minicom*. La sintaxis de la opción del *minicom* es de la siguiente manera:

```
$ sudo su
$ minicom openwrtalix
```

Con *minicom openwrtalix* se establece conexión, siempre y cuando *minicom* esté bien configurado. Para configurar el *minicom* se debe ejecutar éste y una vez dentro del programa:

- ctrl+a:
  - opción 'o'
  - opción 'f' (para poner el control de flujo por hardware=*NO* (si se deja en *SI* no deja escribir ninguna orden por el *prompt*).
- Velocidad Alix 2C0: 38400

[2] NOTA: se debe estar muy seguro de la unidad que se formatea ya que se puede dañar la información del disco duro del PC.

- Puerto de conexión: ttyS0

3.2.2. OpenWRT para las computadoras embebidas Pronghorn Metro SBC, Pronghorn SBC 250 y Avila GW2348-4 (XScale)

A continuación se enumeran los cambios de configuración (en el *make menuconfig*) para las computadoras embebidas *XScale*. Esta configuración permite sacar los archivos que se deberán transferir a los equipos:

```
Target System: Intel IXP4xx
Subtarget: Generic
Target Profile: Default Profile
Target Images
          jffs2: N
          ext2: N
Network
          hostapd: M
          wpa-supplicant: M
Kernel modules
          Filesystems
kmod-fs-ext3: M
          Network Devices
                    kmod-via-rhine: *
                    TODOS LOS DEMÁS...: N
          USB Support
                    kmod-usb-core: M
                              kmod-usb-ohci: M
                              kmod-usb-storage: M
                              kmod-usb2: M
          Wireless Drivers
                    kmod-madwifi:

          Utilities
                    disc
```

```
cfdisk: M
swap-utils: M
```

```
$ make world
```

Se ejecuta el *make world* y se crea el *firmware*. Los archivos generados se guardan dentro de la carpeta */bin*. Para instalar el sistema operativo en las computadoras embebidas debemos enviarles dos archivos; el primero es '*openwrt-XXX-zImage*' y el segundo '*openwrt-ixp4xx-squashfs.img*'. El primer archivo varia según el fabricante, el segundo es común para la misma arquitectura *XScale*.

Los dos archivos correspondientes se transferirán a las memorias de las computadoras embebidas por red, mediante una conexión *TFTP*. Para ello se debe usar un *servidor TFTP*. No todos los servidores *TFTP* son compatibles, en estos casos se ha usado el *atftpd*.

Cabe destacar que en el archivo *'/etc/default/atftpd'* se asigna una carpeta por defecto llamada */tftpboot*, que es donde se deben guardar los archivos que tienen que ser transferidos a las computadoras embebidas. Por tanto se debe crear esta carpeta [3].

Para que exista comunicación entre PC y computadora embebida se debe configurar la red. Hay que poner una *IP* y una máscara al *PC* y también configurar la red de la computadora embebida. Para configurar la *IP* de la computadora embebida se debe ejecutar *fconfig* en el *redboot*, y con esta orden aparecen las opciones a configurar.

El siguiente paso es ver la instalación del *OpenWRT* en la computadora embebida *Avila GW2348-4 Network Platform* (ver *datasheet* en anexo **A.1**).

Instalación del kernel OpenWRT para la computadora embebida Avila GW2348-4 Network Platform

Para acceder vía cable serie a la computadora embebida *Avila GW2348-4*, y poder ejecutar las ordenes, se hará mediante el *minicom*:

```
$ sudo su
```

[3] http://www.ubuntu-es.org/index.php?q=node/35185

```
$ minicom openwrtavila
```

Se debe configurar el *minicom*:

- ctrl+a:
  - opción 'o'
  - opción 'f' (para poner el control de flujo por hardware=*NO* (si se deja en *SI* no deja escribir ninguna orden por el *prompt*).
- Velocidad Avila GW2348-4: 115200
- Puerto de conexión: ttyUSB0

Se conecta la computadora embebida con el PC y en un terminal de esta se abre la conexión serie. La *IP* que viene por defecto en la *Avila GW2348-4* es *192.168.3.2*:

```
== Executing boot script in 2.500 seconds - enter ^C to abort
```

Se tienen dos segundos para hacer *ctrl+c'* y después se debe ver un *prompt* como el siguiente:

```
^C
RedBoot>
```

Se entra la siguiente linea:

```
fis init
```

Y la GW2348-4 responderá con lo siguiente:

```
About to initialize [format] FLASH image system - continue (y/n)?
```

Se acepta con *Y* y la computadora embebida responde así:

```
*** Initialize FLASH Image System
... Erase from 0x50080000-0x50fe0000: ..............
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
```

```
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
RedBoot>
```

Aquí se debe ejecutar la siguiente linea (el archivo *openwrt-avila-zImage* debe encontrarse dentro de la carpeta */tftpboot*):

```
load -r -v -b 0x00800000 openwrt-avila-zImage
```

Si el servidor *TFTP* no esta bien configurado, por ejemplo las *IP*'s mal puestas, se obtendrá una respuesta como la siguiente:

```
Using default protocol (TFTP)
__udp_sendto: Can't find address of server
Can't load 'zImage': some sort of network error
RedBoot>
```

Con el servidor *TFTP* bien configurado se ve lo siguiente:

```
Using default protocol (TFTP)
/
Raw file loaded 0x00800000-0x00967c93, assumed entry at 0x00800000
RedBoot>
```

Se espera varios segundos a que se cargue el archivo y con el siguiente comando se empieza la instalación:

```
fis create linux
```

La respuesta debería ser esta:

```
... Erase from 0x50080000-0x50280000: ..............
... Program from 0x00800000-0x00a00000 at 0x50080000: ..............
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
RedBoot>
```

Ahora se debe cargar el archivo de *rootfs* pero primero se observa cuanto espacio libre queda en la memoria:

```
RedBoot> fis free
  0x50180000 .. 0x50FE0000
```

Se tiene *0xE60000* de espacio libre. Se carga el segundo archivo:

```
load -r -v -b 0x00800000 openwrt-ixp4xx-squashfs.img
```

Se obtiene la siguiente respuesta:

```
Using default protocol (TFTP)
/
Raw file loaded 0x00800000-0x00d13fff, assumed entry at 0x00800000
RedBoot>
```

Después de que cargue el archivo se inicia la instalación:

```
fis create -l 0xe60000 rootfs
```

Programar esto tomará bastante tiempo, debido a tu tamaño. La respuesta será la siguiente:

```
... Erase from 0x50180000-0x50fe0000: ..............
... Program from 0x00800000-0x00920000 at 0x50180000: ..............
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
```

Una vez terminado, se ejecuta la utilidad *fconfig* en el *prompt* de *reedboot*.

```
RedBoot> fconfig
Run script at boot: true
Boot script:
.. fis load ramdisk
.. fis load zimage
```

```
.. exec
Enter script, terminate with empty line
>> fis load linux
>> exec
>>
Boot script timeout (100ms resolution): 25
Use BOOTP for network configuration: false
Gateway IP address:
Local IP address: 192.168.3.2
Local IP address mask: 255.255.255.0
Default server IP address: 192.168.3.1
Console baud rate: 115200
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Default network device: npe_eth0
Update RedBoot non-volatile configuration - continue (y/n)? y
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
RedBoot>
```

Ahora se debe resetear la unidad y la computadora embebida ya iniciará el *OpenWRT*:

```
reset
```

A continuación se verá la instalación del *OpenWRT* en las dos computadoras embebidas *Pronghorn*, SBC 250 y Metro SBC. Cabe destacar que el *kernel* para estas dos computadoras embebidas será el mismo debido a la similitud de algunas de sus características.

Instalación del kernel OpenWRT para las computadoras embebidas Pronghorn metro SBC y Pronghorn SBC 250

Para acceder vía cable serie a las computadoras embebidas *Pronghorn* y poder ejecutar las ordenes se hará mediante el *minicom*:

```
$ sudo su
$ minicom openwrpronghorn
```

Se debe configurar el *minicom*. Ejecutar '*ctrl+a*' y la opción '*o*'; Para estos dos modelos de *Pronghorn* se debe configurar el *minicom* con una velocidad de transmisión de *115200*, el puerto de conexión *ttyUSB0*, y con la opción '*F*', poner el control de flujo por hardware=*NO* (si se deja en *SI* no deja escribir ninguna orden por el *prompt* de *minicom*).

Con el servidor *TFTP* ya configurado, se conecta la computadora embebida y, ya se puede empezar el proceso de transferencia del sistema operativo mediante un terminal *Linux*.

Igual que en el caso anterior, de la computadora embebida *Avila GW2348-4*, se tienen dos segundos para hacer *ctrl+c'* y después se debe ver un *prompt* como el siguiente:

```
== Executing boot script in 2.500 seconds - enter ^C to abort
^C
RedBoot>
```

Aquí es donde se empiezan a mandar las ordenes de ejecución para transferir los archivos:

```
RedBoot> fis unlock -f 0x50000000 -l 0x1000000
... Unlock from 0x50000000-0x51000000: ..............

RedBoot> fis init -f
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
... Erase from 0x50060000-0x50fc0000: ..............
... Erase from 0x50fe0000-0x50fe0000:
... Unlock from 0x50fe0000-0x51000000: .
```

```
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .

RedBoot> fis list
Name               FLASH addr  Mem addr    Length      Entry point
RedBoot            0x50000000  0x50000000  0x00060000  0x00000000
RedBoot config   0x50FC0000 0x50FC0000  0x00001000  0x00000000
FIS directory       0x50FE0000  0x50FE0000  0x00020000  0x00000000

RedBoot> load -r -v -b \%{FREEMEMLO} -h 20.20.20.80 openwrt-pronghorn-zImage
CCCCRaw file loaded 0x00029c00-0x00029bff, assumed entry at 0x00029c00
xyzModem - CRC mode, 0(SOH)/0(STX)/0(CAN) packets, 20 retries

RedBoot> fis cre linux
... Erase from 0x50060000-0x50060000:
... Program from 0x00029c00-0x00029c00 at 0x50060000:
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .

RedBoot>load -r -v -b \%{FREEMEMLO}-h 20.20.20.80 openwrt-ixp4xx-squashfs.img
CCCCRaw file loaded 0x00029c00-0x00029bff, assumed entry at 0x00029c00
xyzModem - CRC mode, 0(SOH)/0(STX)/0(CAN) packets, 20 retries

RedBoot> fis cre -l 0xe60000 rootfs
... Erase from 0x50060000-0x50ec0000: ..............
... Program from 0x00029c00-0x00029c00 at 0x50060000:
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x03fe0000-0x04000000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .

RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0x50000000  0x50000000  0x00060000  0x00000000
linux             0x50060000  0x00029C00  0x00000000  0x00029C00
RedBoot config    0x50FC0000  0x50FC0000  0x00001000  0x00000000
```

```
FIS directory     0x50FE0000  0x50FE0000  0x00020000  0x00000000

RedBoot> fc -i
Initialize non-volatile configuration - continue (y/n)? y
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis unlock -f 0x50060000 -l 0xfa0000
>> fis load linux
>> exec
>>
Boot script timeout (1000ms resolution): 2
Use BOOTP for network configuration: false
Gateway IP address: 0.0.0.0
Local IP address: 192.168.3.2
Local IP address mask: 255.255.255.0
Default server IP address:  192.168.3.1
Console baud rate: 115200
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Default network device: npe_eth0
Network hardware address [MAC] for NPE eth0: XXXX
Network hardware address [MAC] for NPE eth1: XXXX
Update RedBoot non-volatile configuration - continue (y/n)? y
... Unlock from 0x50fc0000-0x50fc1000: .
... Erase from 0x50fc0000-0x50fc1000: .
... Program from 0x03fd3000-0x03fd4000 at 0x50fc0000: .
... Lock from 0x50fc0000-0x50fc1000: .
RedBoot>
```

3.3. Configuración por comandos de un enlace Wi-Fi en OpenWRT

Con las siguientes ordenes se puede configurar las interfaces de red en un sistema con *OpenWRT*, el único inconveniente es que esta configuración se perderá si se

reinicia. Las tarjetas inalámbricas necesarias se deben conectar antes de alimentar la computadora embebida.

Con las siguientes lineas se crea una red inalámbrica *wifi0* en la interfaz *ath0*, en modo cliente *STA*, con nombre de red *OPENWRT* y que opera en el *canal 5*. Si se quisiera configurar en modo *AP* tan solo se tendría que cambiar el *sta* de la primera linea por *ap*:

```
$ wlanconfig ath0 create wlandev wifi0 wlanmode sta
$ iwconfig ath0 essid OPENWRT channel 5
```

Con la siguiente linea se obliga a usar el *modo 3* que corresponde al *802.11 g*.

```
$ iwpriv ath0 mode 3
```

Se configura la *IP* y la máscara de red.

```
$ ifconfig ath0 20.20.20.210 netmask 255.255.255.0
```

Se escanea el canal para ver que redes se captan y se hace *ping* al *AP*, al cual se está colgado.

```
$ iwlist ath0 scan
$ ping 20.20.20.203
```

3.4. Configuración de enlaces Wi-Fi en OpenWRT por medio de archivos de configuración

Obviamente *OpenWRT* permite muchas configuraciones y también muchas opciones de configuración. En esta sección tan solo se hace un pequeño resumen de los archivos que se deben modificar para poder tener una configuración básica de las redes de las que disponen las alimentar la computadoras embebida que se usarán.

Los archivos en *OpenWRT* que se deben modificar para configurar las redes se encuentran dentro de */etc/config*, y son tres; *network*, *wireless* y *dhcp*. Se empieza analizando el archivo *network* viendo un ejemplo:

```
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.25'
option 'netmask' '255.255.255.0'
option 'gateway' '192.168.238.1'
#----------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 50.50.50.2
option netmask 255.255.255.0
#----------------------------
config 'interface' 'wan1'
option ifname ath1
option proto static
option ipaddr 60.60.60.1
option netmask 255.255.255.0
```

En el archivo anterior se puede ver que se configuran varias redes. Destacar el punto *option 'proto' 'static'*, si se quiere configurar como *dhcp*, se debe substituir en lugar de *static* y ya no hace falta que se definan la *IP* y la *máscara*. También se puede ver la forma como se definen las redes inalámbricas, *wan0* y *wan1*. En el ejemplo solo se definen dos redes pero se pueden definir tantas como soporte la computadora embebida.

En el siguiente archivo, */etc/config/wireless*, se definen las características de las redes *wan0* y *wan1*. Como se ve en las lineas siguientes se definen el tipo de tarjeta que se

usará, el canal, la distancia de comunicación, el modo, el *ssid* o la encriptación, ente otros campos. También se define si la red actuará como *AP* o como *STA*. La primera parte del archivo define la *wifi0* y la segunda la *wifi1*:

```
config wifi-device wifi0
option type atheros
option channel 6
#option distance 100
option disabled 0
#
config wifi-iface
option device wifi0
option network wan0
option mode sta
option ssid op5
#option encryption wep
#option hidden 0
#option key 1
#option key1 0123456789
#----------------------------
config wifi-device wifi1
option type atheros
option channel 4
#option distance 100
option disabled 0
#
config wifi-iface
option device wifi1
option network wan1
option mode ap
option ssid op6
#option encryption wep
#option hidden 0
#option key 2
#option key2 9876543210
```

El siguiente archivo es se encuentra */etc/config/dhcp*, y es el archivo en el cual se pueden modificar los parámetros del *dhcp* de las interfaces.

```
config dnsmasq
option domainneeded 1
option boguspriv 1
option filterwin2k '0'  #enable for dial on demand
option localise_queries 1
option local '/lan/'
option domain 'lan'
option expandhosts 1
option nonegcache 0
option authoritative 1
option readethers 1
option leasefile '/tmp/dhcp.leases'
option resolvfile '/tmp/resolv.conf.auto'
#list server '/mycompany.local/1.2.3.4'
#option nonwildcard 0
#list interface br-lan
#-----------------------------
config dhcp lan
option interface lan
option start 100
option limit 150
option leasetime 12h
#-----------------------------
config dhcp wan
option interface wan
option ignore 1
```

Capítulo 4: Pruebas en el laboratorio

4.1. Montaje de la red de pruebas en el laboratorio

La siguiente red es para hacer las pruebas de nivel de señal y relación de la calidad del enlace, y medir el rendimiento de los sistemas. La red implementada **(ver fig. 4.1)** servirá para hacer pruebas entre tres combinaciones; El primero será un enlace entre una computadora embebida *Pronghorn Metro SBC* configurada como *AP* y una *Pronghorn Sbc 250* configurada como *STA*. El segundo entre dos computadoras embebidas *Avila GW2348-4*. Y el tercero entre dos *Alix 2C0*.

Seguidamente se ve el modelo de red que se configurará para los enlaces:

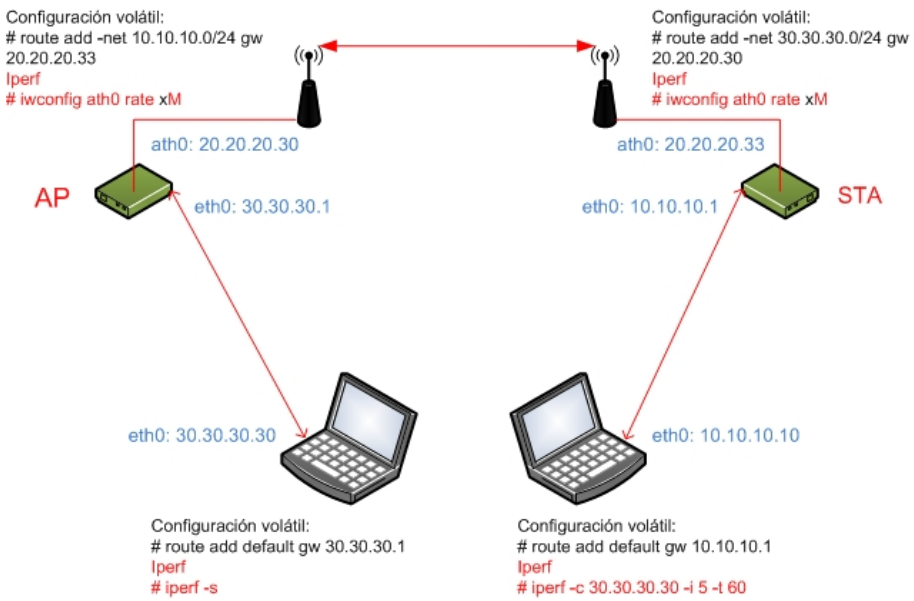


Figura 4.1: Configuración de la red para las pruebas de nivel de señal y relación de calidad de los enlaces.

Se usan los dos PC de los extremos, donde se ha instalado *iperf*, para realizar la prueba de rendimiento.

4.1.1. Prueba de nivel de señal y relación de la calidad del enlace

Esta prueba se hace mediante el comando *# iwconfig*. Así se muestra por pantalla, entre otros, los tres valores necesitados: *Link Quality*, *Signal Level* y *Noise Level*. Para tener un valor más real de los niveles, se tomaron diez valores en instantes de tiempo diferentes (la diferencia entre valores era mínima) y se hizo la media:

Cuadro 4.1: Tabla de valores obtenidos con el *iwconfig*.

Nivel de señal y relación de calidad del enlace	Link Quality [%]	Signal Level [dBm]	Noise Level [dBm]
Metro SBC (AP) y SBC 250 (STA)	65/70	-31 dBm	-96 dBm
Enlace entre dos Avila GW2348-4	67/70	-30 dBm	-97 dBm
Enlace entre dos Alix 2C0	64/70	-32 dBm	-96 dBm

4.1.2. Prueba de rendimiento del enlace

Tal y como ya se especifica en la imagen **(ver fig. 4.1)** esta prueba se realiza mediante el comando *# iperf*. Su configuración también está definida en cada punto de la imagen; En un PC se hace *iperf -c* y en el otro *iperf -s*. El PC de *iperf -s* se pone a la escucha en un puerto, y ahí es donde *iperf -c* emite los señales de prueba. En *iperf -c*, el *-i 5* hace referencia a los segundos de intervalo entre cada muestra y, el *-t 60* son los segundos que dura la prueba. En el AP y el STA, el *xM* representa la velocidad máxima de transmisión (*6M, 12M, 18M, 36M, 54M*), que se irá modificando. El resultado de la prueba será en *Mbits/sec* y sale al final de la prueba en el terminal de *iperf -s*. La siguiente tabla refleja los valores obtenidos:

Cuadro 4.2: Tabla de resultados de *iperf*.

Medición del rendimiento del enlace	6M	12M	18M	36M	54M
Metro SBC (AP) y SBC 250 (STA)	4.41 Mbits/s	8.48 Mbits/s	13.0 Mbits/s	24.2 Mbits/s	32.5 Mbits/s
Enlace entre dos Avila GW2348-4	4.39 Mbits/s	8.60 Mbits/s	13.0 Mbits/s	24.3 Mbits/s	34.3 Mbits/s
Enlace entre dos Alix 2C0	4.38 Mbits/s	8.60 Mbits/s	13.1 Mbits/s	24.2 Mbits/s	34.2 Mbits/s

4.1.3. Prueba de consumo de energía

Para analizar el consumo de energía de cada computadora embebida se considerarán varios casos, que se pueden ver en la tabla que viene a continuación. Para calcular la potencia que se consume, ya sabiendo el voltaje que usa cada equipo, sólo hace falta medir los amperios que se consumen en cada momento (mediante un amperímetro y una fuente adaptada para hacer éste calculo).

Observaciones:

- Para medir la corriente se usó un amperímetro digital conectado a un osciloscopio.
- Las medidas se han tomado usando tarjetas *SR2*, exceptuando las pruebas con la computadora embebida *Pronghorn Metro SBC* que se hicieron con tarjetas *XR2*, cosa que también se especifica en la tabla.
- El consumo de energía del puerto serie en el momento del acceso es muy bajo, a lo más será de *0.1Watts*.
- En la siguiente tabla C.E. significa computadora embebida
- Se pueden ver las fotos de la prueba en el anexo **A.4**.

Cuadro 4.3: Consumo de energía de las computadoras embebidas [Amperios].

Consumo de energía:	Avila GW2348-4 12V	Alix 2C0 12V	SBC 250 5V	Metro SBC 48V	Metro SBC (con XR2) 48V
C.E. sin tarjetas	0.33 A	0.24 A	0.29 A	0.08 A	0.08 A
C.E. con 1 tarjeta sin configurar	0.36 A	0.28 A	0.34 A	0.09 A	0.09 A
C.E. con 2 tarjetas sin configurar	0.38 A	0.30A	0.36 A	0.10 A	0.10 A
C.E. con 3 tarjetas sin configurar	0.40 A	-	-	0.11 A	0.11 A
C.E. con 4 tarjetas sin configurar	0.42 A	-	-	0.12 A	0.12 A
C.E. con 1 tarjeta configurada	0.45 A	0.37 A	0.52 A	0.11 A	0.11 A
C.E. con 2 tarjetas configuradas	0.57 A	0.47 A	0.73 A	0.13 A	0.13 A
C.E. con 3 tarjetas configuradas	0.70 A	-	-	0.15 A	0.15 A
C.E. con 4 tarjetas configuradas	0.82 A	-	-	0.17 A	0.17 A
C.E. con 1 tarjeta Tx, sin tarjetas vecinas	0.63 A	0.49 A	1.04 A	0.13 A	0.18 A
C.E. con 1 tarjeta Rx, sin tarjetas vecinas	0.50 A	0.38 A	0.54 A	0.11 A	0.11 A
C.E. con 2 tarjetas Tx, sin tarjetas vecinas	0.75 A	0.59 A	1.28 A	0.17 A	0.23 A
C.E. con 2 tarjetas Rx, sin tarjetas vecinas	0.63 A	0.48 A	0.76 A	0.12 A	0.12 A
C.E. con 1Tx y 1Rx, sin tarjetas vecinas	0.72 A	0.58 A	1.06 A	0.14 A	0.19 A
C.E. sin tarjetas y eth0 Tx	0.40 A	0.32 A	0.38 A	0.10 A	0.10 A
C.E. sin tarjetas y eth0 Rx	0.40 A	0.32 A	0.38 A	0.09 A	0.09 A

4.1.4. Prueba de potencia de transmisión de las tarjetas

Mediante el analizador de espectros se tomarán los valores reales de potencia de transmisión de las tarjetas ya comentadas (*Sr2*, *R52h*, *Engenius* y *XR2*). Es bueno tener estos valores para poder compararlos con los valores teóricos provistos por el fabricante.

Cuadro 4.4: Tabla de valores de las potencias de transmisión de las tarjetas.

Ubiquiti Networks Super Range 2 (SR2)	26.31 dBm - 27.46 dBm
Mikrotik R52H	23.19 dBm - 22.77 dBm
Ubiquiti Networks Xtreme Range 2 (XR2)	27.98 dBm - 29.27 dBm
Engenius EMP-8602	26.31 dBm - 26.64 dBm

4.2. Conclusiones y observaciones de las pruebas en el laboratorio

Después de finalizar todas las pruebas en el laboratorio se hace una valoración general de los resultados y se prueba si los sistemas satisfacen las necesidades planteadas.

Si se refiere a los niveles de señal, ruido y calidad del enlace no hay nada que objetar, se obtienen valores muy buenos. Cabe recordar que las pruebas son en laboratorio, no son pruebas en un escenario real donde los valores diferirán mucho de los resultados aquí obtenidos.

En las pruebas de rendimiento se puede ver que los resultados siempre están un poco por debajo de los valores esperados teóricamente. A pesar de ello, los valores empíricos obtenidos siguen siendo muy buenos. Es bueno tener en cuenta que son enlaces de laboratorio, por eso mismo los valores de un enlace real seguramente estarán muy lejos de los valores obtenidos.

La prueba de consumo es la que más se puede asemejar a la realidad ya que el consumo en campo es muy parecido al consumo obtenido al laboratorio. Si se toma los doce voltios como referencia, se puede ver que la corriente media es de unos *0,6 A*, teniendo picos de *0,80 A*, eso implica que el consumo medio es de unos *7.2 W* con

picos de hasta *9.6 W*. Estos valores son alcanzables por sistemas fotovoltáicos implementados con paneles y baterías. Normalmente el GTR usa sistemas fotovoltáicos que generan hasta *75 W* y baterías que garantizan energía para dos días, en caso de que el sistema se malogre.

La prueba de la potencia de transmisión se hizo sobretodo para establecer una comparación entre los valores reales obtenidos y los valores teóricos de cada tarjeta. Con todo ello, si se compara estos valores teóricos, vistos en el análisis de las tarjetas, con los valores obtenidos de las pruebas, se puede ver que no varían mucho.

A modo resumen de las conclusiones, decir que los valores obtenidos en estas pruebas certifican la idea que ya se tenia a priori de que estas combinaciones de hardware y software *OEM* podrían ser usadas para telecomunicaciones rurales.

Capítulo 5: Pruebas en el campus universitario

5.1. Diseño de la red e implementación en el laboratorio

Con el objetivo de medir el rendimiento de las soluciones encontradas, se configurará una red privada de comunicación inalámbrica (siete enlaces) que será analizada por varios *scripts* **[9]** que recopilarán información durante un tiempo determinado (seis días). La red estará configurada por cinco puntos entrelazados y se sitúa dentro del campus universitario **(ver anexo A.5)**.

La red de pruebas se instaló en el campus de la Pontificia Universidad Católica del Perú, en los edificios más altos y/o ubicados estratégicamente. Esta red permite hacer varias pruebas sobre comunicaciones inalámbricas con la diferencia de que, si se compara con una red de laboratorio, permite tener un escenario de pruebas más parecido a la realidad.

Antes de montar la red en sus puntos, con las antenas correspondientes y la configuración propia de campo, se configurarán los equipos en el laboratorio, se probará la red y se dejarán todos los *programas* que se necesitarán para recopilar la información. Los equipos se dejarán configurados y probados para evitar fallos. El diagrama de la figura **5.1** muestra las configuraciones. La red constará de cinco puntos y cada punto se configurará con una computadora embebida y dos o cuatro tarjetas de red, dependiendo de los enlaces necesarios en cada punto.

En una primera instancia se instaló la computadora embebida *Pronghorn Metro SBC* en uno de los puntos y se hicieron varias pruebas, pero al arrancar no detectaba las interfaces inalámbricas, cada vez que se iniciaba detectaba una, dos, tres o cuatro interfaces aleatoriamente. Así pues, debido a su poca estabilidad en el funcionamiento, se decidió substituirla por una más robusta en condiciones de software y se usó otra *Avila GW2348-4*.

Cada computadora embebida contará con *OpenWRT* instalado con en controlador *madwifi* para poder usar las tarjetas inalámbricas. También se dotará los equipos de los *scripts* necesarios para el análisis de los enlaces. Habrá un *script*, dedicado ha recopilar información de *ping* en un enlace, así como niveles de señal y ruido, y otros dos *scripts* enfocados a monitorizar información de rendimiento.

Para entrar en el análisis de estos *scripts* primero se debe aclarar la configuración de

la red, interfaces, IPs, etc. La siguiente imagen representa la configuración final de la red que se usará para las pruebas:

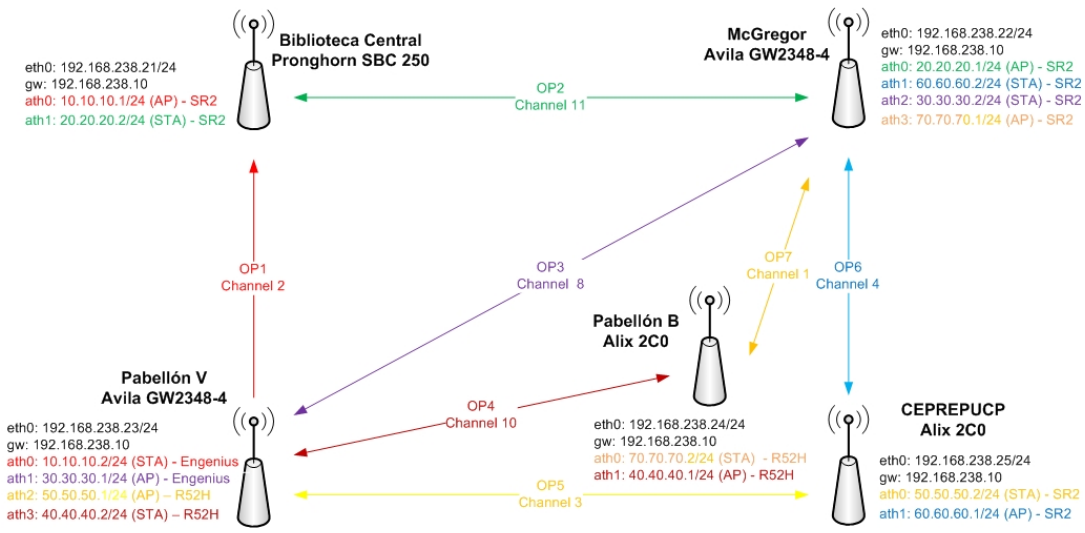


Figura 5.1: Configuración de la red de pruebas.

Cada enlace esta montado con un *AP* (punto de acceso) y un *STA* (cliente) y se ha asignado un *ESSID* con el nombre *OPX* (donde X es el número de enlace). En la imagen también se puede ver el canal que se esta usando para cada enlace, así como las *IP's* correspondientes a cada tarjeta inalámbrica.

Con la configuración de la imagen anterior se monta la red en el laboratorio y se hacen las primeras pruebas de funcionamiento. Tanto los *scripts* que se usan como la configuración final de la red van cambiando durante las pruebas para optimizar su funcionamiento.

La siguiente imagen muestra los equipos de la red funcionando en el laboratorio:

Figura 5.2: Imagen de la red funcionando en el laboratorio.

5.2. Configuración de los equipos para las pruebas

Como ya se ha comentado, a cada equipo se le modificarán los archivos correspondientes y se le instalarán los paquetes necesarios para que cumpla con todas las funciones que se requieren. A continuación se analizará la configuración de un sistema genéricamente así como los paquetes que se le han instalado.

Es indispensable diferenciar entre computadoras embebidas de arquitectura *X86* y *XScale*. Para las diferentes arquitecturas se deben bajar los paquetes correspondientes. En el caso de estas pruebas se instaló a los equipos el *ntpdate*, el *iperf* y *uclibcxx* (necesario para *iperf*). Estos paquetes se obtuvieron de la pagina de OpenWRT [1] (*ntpdate_4.2.4p7-1_XXX.ipk*, *iperf_2.0.4-2_XXX.ipk* y *uclibcxx_0.2.2-2_XXX.ipk*).

El paquete *ntpdate* es para que un equipo sincronice su hora en función de un servidor (*NTP*). Se debe descargar el paquete, transferirlo a la computadora embebida y, desde el mismo directorio donde se haya transferido el paquete, instalarlo mediante *opkg*. La orden será del estilo *"$ opkg install ntpdate_4.2.4p7-1_XXX.ipk"*, donde *"XXX"* se

[1] http://downloads.openwrt.org/snapshots/trunk/

deberá substituir por *x86* o *ixp4xx* según la arquitectura. Se utiliza el comando *ntpdate -u -b 200.16.6.30* antes de recoger los datos de los enlaces.

Para la prueba de rendimiento se instala *iperf* en los quipos, donde *"XXX"* se deberá substituir por *x86* o *ixp4xx* según la arquitectura:

```
$ opkg install iperf_2.0.4-2_XXX.ipk
$ opkg install uclibcxx_0.2.2-2_XXX.ipk
```

Para configurar la zona horaria del equipo es necesario que se modifique el archivo *boot* que se encuentra en */etc/init.d/*, substituyendo las tres siguientes lineas:

```
config_get timezone "$cfg" timezone 'UTC+5'
#echo "$timezone" > /tmp/TZ
echo "UTC+5" > /tmp/TZ
```

5.2.1. Configuración de los enlaces

La configuración lógica de cada enrutador y sus enlaces se hace mediante dos archivos de configuración y un *script* que se ejecuta cuando arranca el equipo. Los archivos se llaman *wireless* y *network* y se encuentran en */etc/config/*. El *script* de nombre */etc/init.d/redes* a la configuración de los enlaces */Wi-Fi*.

Las siguientes lineas son el contenido del archivo *wireless* para la configuración de un enlace. Para cada enlace se deben especificar los mismos parámetros, así si se tienen dos enlaces se deben duplicar estas lineas.

```
config wifi-device wifi0
option type atheros
option channel X
option disabled 0
#----------------------------
config wifi-iface
option device wifi0
option network wan0
option mode ap
```

```
option ssid opX
```

Como se puede ver, es en este archivo donde se especifica el *canal* por donde se trasmitirá, el *modo*, el *ssid*, etc. Los demás parámetros van especificados en el archivo *network* que se puede ver a continuación:

```
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.XX'
option 'netmask' '255.255.255.0'
#----------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 50.50.50.1
option netmask 255.255.255.0
```

En este archivo se especifica primero la configuración de la interfaz de *loopback*, después la *ethernet* con su *IP* y su máscara y finalmente las interfaces inalámbricas. En este caso, si se quiere añadir otra interfaz inalámbrica solo se deben duplicar y configurar las ultimas cinco lineas, en ellas es donde se especifica la configuración lógica de cada enlace y su correspondiente *ath*. Como se ve es donde se especifica, entre otras cosas, las *IP's* que se usan, y si son estáticas o dinámicas, y la máscara de red.

Cuando arranca el equipo debe ejecutar el *script redes*. Para que el *script* se pueda ejecutar se debe primero cambiar los permisos al archivo, y segundo poner la orden de ejecución en el arranque.

```
chmod 744 redes
ln -s /etc/init.d/redes /etc/rc.d/S90redes
```

El archivo redes, como se puede analizar a continuación, modifica primero la potencia de transmisión de cada enlace, segundo limita la velocidad de transmisión, tercero añade a la tabla de rutas el *gateway* por defecto y finalmente también modifica la diversidad de la antena en función de la que se use (cada tarjeta inalámbrica dispone de dos antenas). La ultima orden del *script*, como ya se ya visto anteriormente, es para sincronizar la fecha y la hora.

```
#!/bin/sh /etc/rc.common
START=90
start() {
iwconfig ath0 txpower 23
iwconfig ath0 rate 6M
route add default gw 192.168.238.10
echo 0 > /proc/sys/dev/wifi0/diversity
echo 2 > /proc/sys/dev/wifi0/txantenna
echo 2 > /proc/sys/dev/wifi0/rxantenna
ntpdate -u -b 200.16.6.30
}
```

5.2.2. Recopilación de datos

Una vez se ha arrancado el equipo con las ordenes anteriores ejecutadas, ya se tiene la red preparada para el análisis. Para ello se usarán tres *scripts* por enlace (*latencia*, *bw-iperf-destinoX1* y *bw-iperf-origenX1*), que en el OpenWRT se almacenarán en la ruta */usr/local/bin/*, que como no existe se debe crear. Como en el *script redes* usado anteriormente, a estos también se debe dar permisos de ejecución mediante *chmod 744 X*. Estos tres *scripts* van sincronizados utilizando el *cron*. Los *scripts* de cada equipo de un enlace quedarán de la siguiente manera:

Equipo A

- /usr/local/bin/bw-iperf-origenX1

Equipo B (quién recopila la información del enlace)

- /usr/local/bin/latencia
- /usr/local/bin/bw-iperf-destinoX1

Se decide analizar información sobre el *ping* porque es importante conocer la *latencia* que tiene el enlace, además de saber su nivel de señal y de ruido. El rendimiento también se analiza ya que es la característica del enlace que da una referencia más real de las posibilidades que se tienen de comunicación.

Para configurar la ejecución de los *scripts* se usa la herramienta *cron*. Se debe crear el archivo */etc/crontabs/* con el siguiente contenido:

```
# Syntax for lines is: minute hour day month dayofweek command #
01 * * * * /usr/local/bin/latencia
56 * * * 1 /usr/local/bin/bw-iperf-destinoX1
57 * * * 2 /usr/local/bin/bw-iperf-origenX2
# X1 y X2 corresponden a redes distintas #
```

Se debe cambiar los permisos del archivo *root* para poder ejecutarlo:

```
chmod 744 root
```

Las lineas del archivo */etc/crontabs/root* del ejemplo anterior son los *scripts* que se usarán para el análisis de cada enlace. Cabe destacar que si un equipo analiza varios enlaces entonces se deberá duplicar el numero de lineas del archivo según el numero de enlaces. Otro punto importante es el momento de ejecución de cada *script*; *latencia* se ejecuta cada hora en el minuto uno, el *script* de destino del rendimiento se ejecuta en el minuto cincuenta y seis y el de origen en el minuto cincuenta y siete, esto se hace para no tener problemas de sincronización entre ellos. Es importante también destacar que el análisis del rendimiento de cada enlace, que implique el mismo equipo, se hará en días de la semana diferentes para evitar problemas con el comando *iperf*. Es por ello que en el apartado *dayofweek* del archivo *root* los *scripts* de origen y destino de redes diferentes se ejecutan en días diferentes.

Seguidamente se puede ver el contenido del *script latencia*:

```
#!/bin/sh
#--------------------------------------------------------------------
# PING
#--------------------------------------------------------------------
ntpdate -u -b 200.16.6.30 &
DATA=0
PLOSS=0
MIN=0
AVG=0
MAX=0
SL=0
NL=0
TMPFILE01=/tmp/tmp.temp01
ping -q X0.X0.X0.1 -w 3250 > $TMPFILE01 #ip opX
DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE01 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $8}')
#--------------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#--------------------------------------------------------------------
iwconfig athN > $TMPFILE01 #athN corresponde a opX
SL=$(cat $TMPFILE01 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE01 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)

#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OPX.txt

ntpdate -u -b 200.16.6.30 &
```

Este *script* obtiene el informe de los *ping* hechos cada hora y los guarda por lineas en */root/result-OPX.txt* además de recopilar información del nivel de señal y de ruido del enlace. Se puede ver que el *script* sincroniza la hora del sistema tanto al principio como al final, esto es para evitar problemas de desfase horario (sobretodo para el *iperf*). La información captada se irá añadiendo cada vez a una linea nueva del archivo */root/result-OPX.txt*.

Al concluir las pruebas, el archivo */root/result-OPX.txt* tendrá una linea como la siguiente por cada hora que haya transcurrido:

```
2010-03-07 03:55  3%  0.951  1.971  110.680  -69  -94
```

Para los *scripts bw-iperf-origenX* y *bw-iperf-destinoX* se necesita usar *iperf*. La función del *script bw-iperf-origenX*, que corresponde al origen de cada enlace, es arrancar el *iperf -c*. Así pues el *script* quedará de la siguiente manera:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 mins
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------
ntpdate -u -b 200.16.6.30 &
iperf -c 60.60.60.2 -i 5 -t 60 -p 5001 #ip origen OP6
ntpdate -u -b 200.16.6.30 &
```

En este caso también se ejecuta *ntpdate -u -b 200.16.6.30* al principio y al final para evitar problemas de desincronización temporal entre equipos. Notar que este *script* se arranca un minuto después del *script* de destino para asegurar que el otro ya esta en funcionamiento cuando este inicie.

Al otro extremo del enlace se debe ejecutar el siguente *script*, el *bw-iperf-destinoX*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------
ntpdate -u -b 200.16.6.30 &
B=0
DATAIPERF=0
TMPFILE11=/tmp/tmp.temp11
DATAIPERF=$(date '+%F %H:%M')
iperf -p 5001 -s > $TMPFILE11 & #iperf -s en bg
```

```
sleep 150
B=$(cat $TMPFILE11 | grep "/sec" | awk '{print $7 $8}')
# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OPX-iperf_DIADELASEMANA.txt
#Guarda el resultado del iperf -s
killall iperf
ntpdate -u -b 200.16.6.30 &
```

Este *script*, que corresponde al destino, se encarga de obtener el rendimiento del enlace cada hora y lo añade al archivo *result-OPX-iperf_DIADELASEMANA.txt*, donde *DIADELASEMANA* se substituye por el nombre del día de la semana que se ejecuta el *script*.

Estos *scripts* se ejecutan un día (laborable) a la semana por enlace. Así, al final de cada día de prueba, se obtendrá un archivo con veinticuatro lineas como la siguiente (una por cada hora del día):

```
2010-03-11 00:56 3.42Mbits/sec
```

Al final de la prueba, para cada enlace que se tenga, se deberá tener dos archivos; */root/result-OPX.txt* y *result-OPX-iperf_DIADELASEMANA.txt*. En función de los resultados de estos archivos se sacarán las estadísticas y conclusiones correspondientes.

Una vez se ha probado la red en el laboratorio, se han hecho diferentes pruebas y se ha comprobado que el funcionamiento es el esperado, ya se puede montar la red en su escenario final **(ver anexo A.5)**.

5.3. Implementación de la red por el campus universitario

Los puntos acordados para crear la red de pruebas dentro de la Pontificia Universidad Católica del Perú, tal y como se puede ver en la imagen **5.1**, son; El edificio Mc Gregor, la biblioteca central, el pabellón B, CEPREPUCP, y el pabellón V.

Con la configuración comentada en la sección anterior **(ver 5.2)**, con la figura **5.1** y con los archivos de configuración adjuntos en el anexo **A.6.1**, ya se conoce la distri-

bución de equipos por los edificios de la PUCP **(ver anexo A.5)** y los archivos que los caracterizan.

Como los sistemas tendrán que estar a la intemperie se montará cada equipo en unas cajas adecuadas para trabajar en el exterior (cajas desarrolladas en Lima y probadas por el GTR). Además, para la prueba dentro del campus universitario, se usarán también las catorce antenas correspondientes (modelo *Hyperlink RE11DP*) para los enlaces, los cables y conectores coaxiales que sean necesarios y los *pigtails* para adaptar la conexión a las tarjetas.

La imagen siguiente muestra el ejemplo de un equipo y sus antenas ya montado en una de las azoteas de la universidad, concretamente el equipo situado en la biblioteca:

Figura 5.3: Equipo correspondiente a la biblioteca.

En el anexo **A.6.3** se muestran más imágenes de los enlaces establecidos.

A continuación se muestran los resultados de las pruebas hechas en el campus de la PUCP. Se debe de tener en cuenta que para las pruebas la velocidad de transmisión se limitó a *6 Mbps*.

5.4. Resultados de las pruebas en el campus universitario

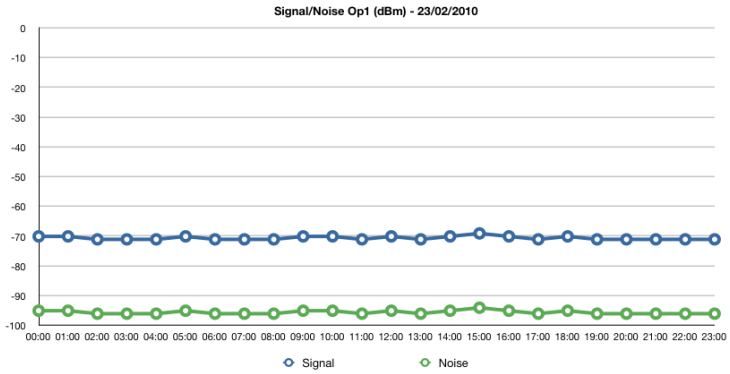


Figura 5.4: Nivel de señal y de ruido en dBm del enlace OP1.

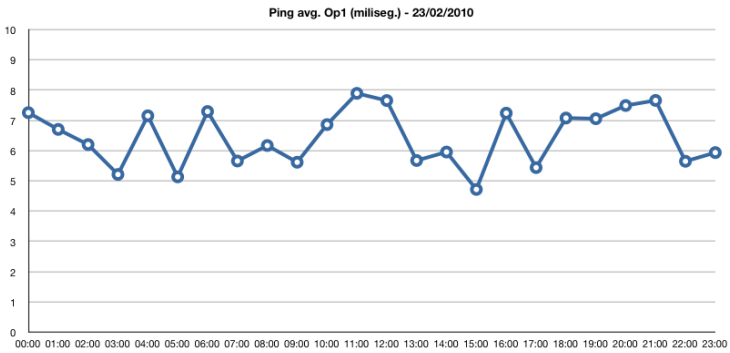


Figura 5.5: Gráfica del valor medio obtenido por el *ping* en el enlace OP1.

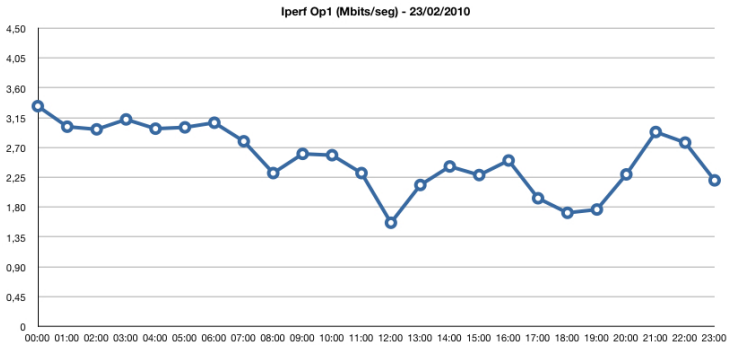


Figura 5.6: Resultado del *Iperf* en el enlace OP1.

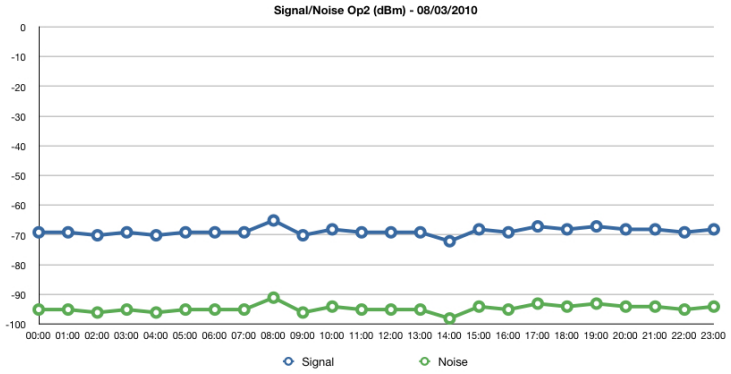


Figura 5.7: Nivel de señal y de ruido en dBm del enlace OP2.

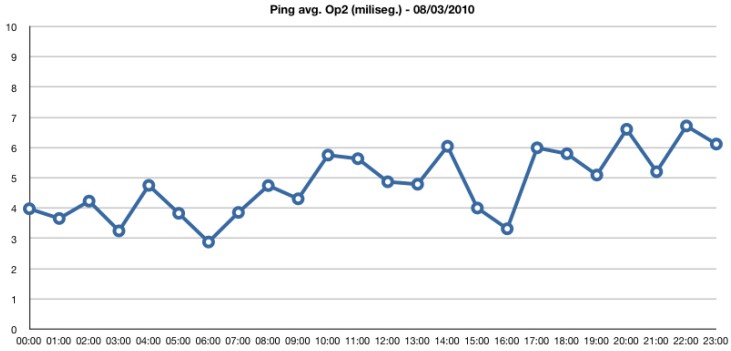


Figura 5.8: Gráfica del valor medio obtenido por el *ping* en el enlace OP2.

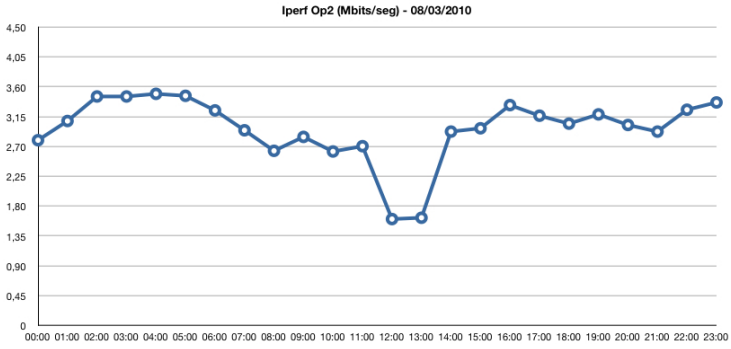


Figura 5.9: Resultado del *Iperf* en el enlace OP2.

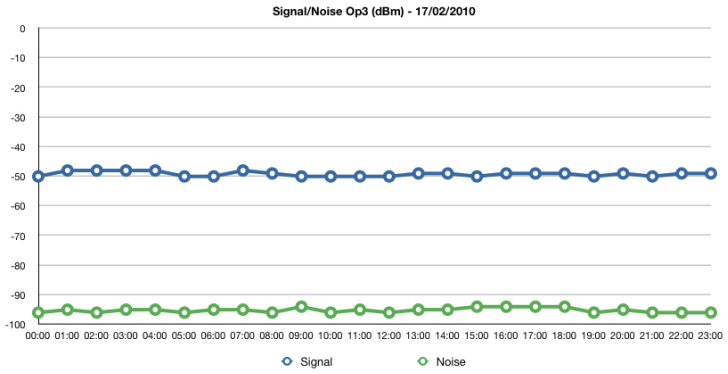


Figura 5.10: Nivel de señal y de ruido en dBm del enlace OP3.

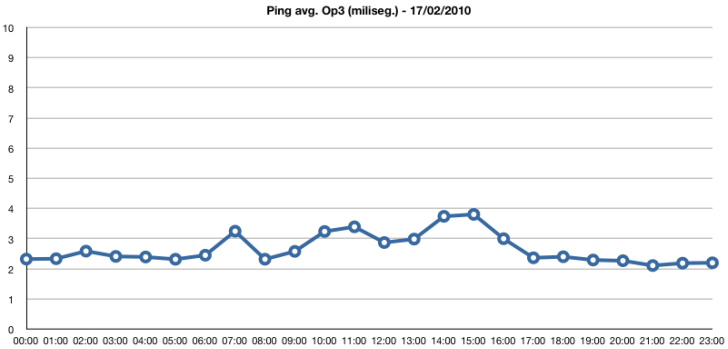


Figura 5.11: Gráfica del valor medio obtenido por el *ping* en el enlace OP3.

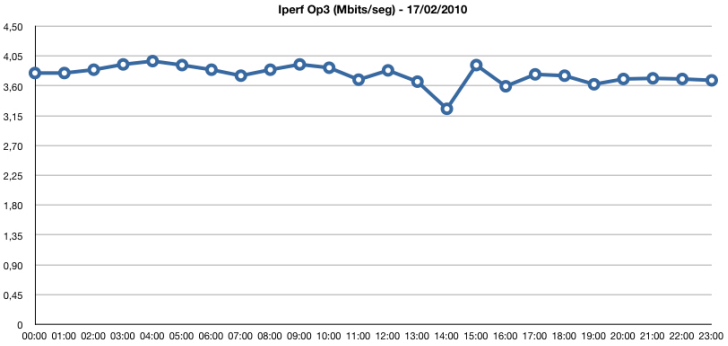


Figura 5.12: Resultado del *Iperf* en el enlace OP3.

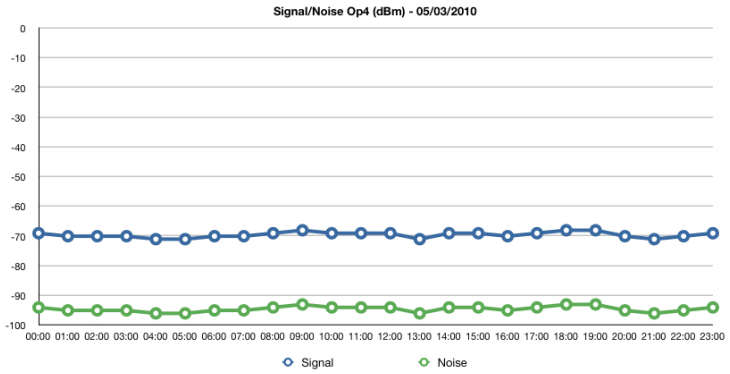


Figura 5.13: Nivel de señal y de ruido en dBm del enlace OP4.

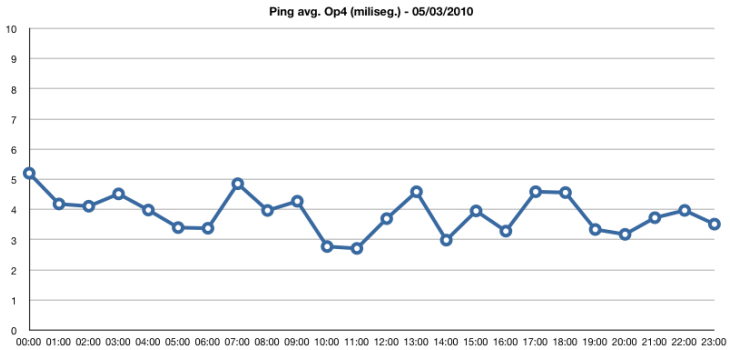


Figura 5.14: Gráfica del valor medio obtenido por el *ping* en el enlace OP4.

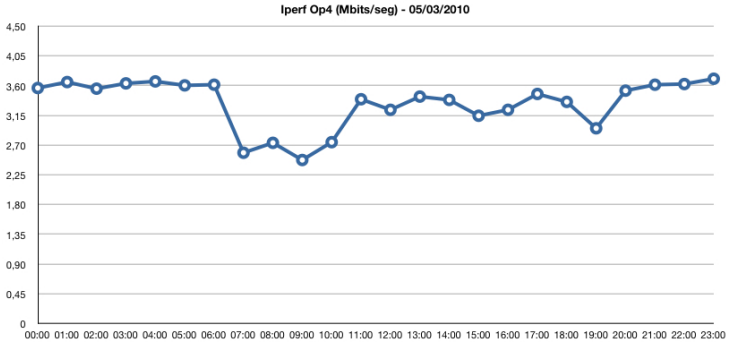


Figura 5.15: Resultado del *Iperf* en el enlace OP4.

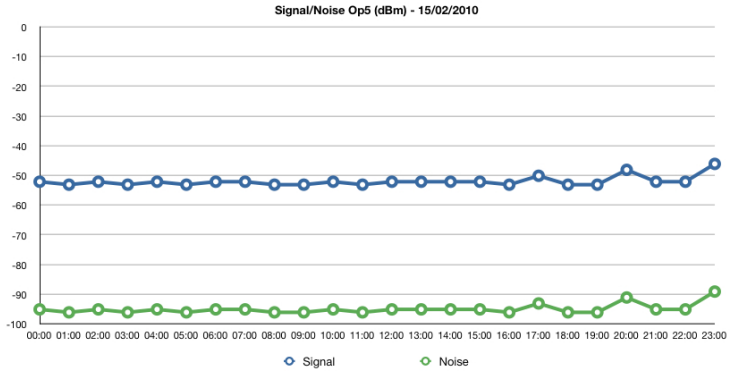


Figura 5.16: Nivel de señal y de ruido en dBm del enlace OP5.

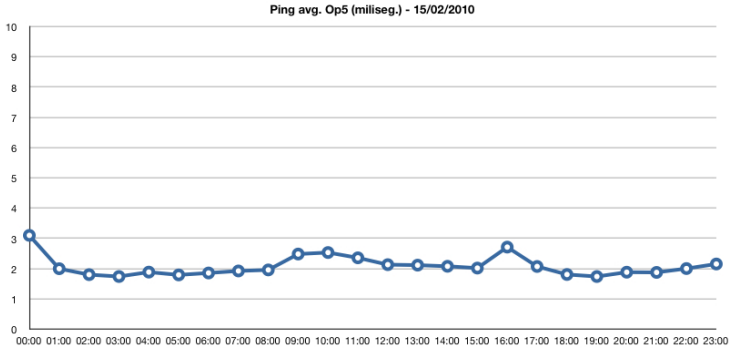


Figura 5.17: Gráfica del valor medio obtenido por el *ping* en el enlace OP5.

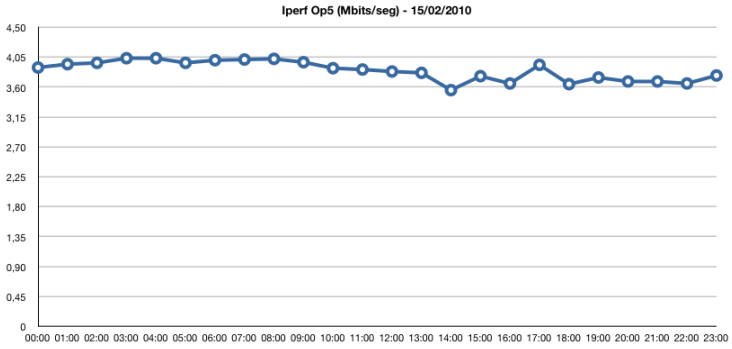


Figura 5.18: Resultado del *Iperf* en el enlace OP5.

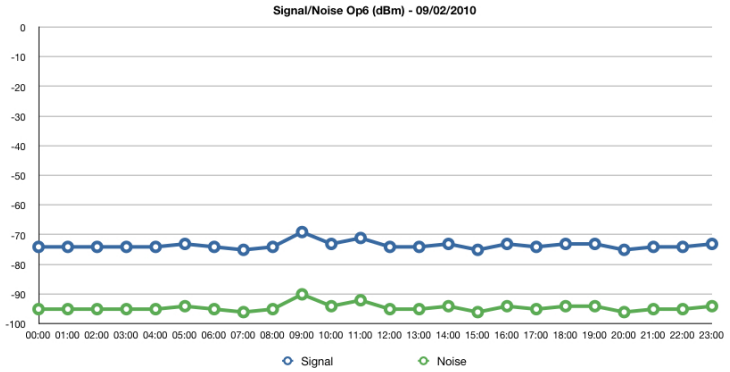


Figura 5.19: Nivel de señal y de ruido en dBm del enlace OP6.

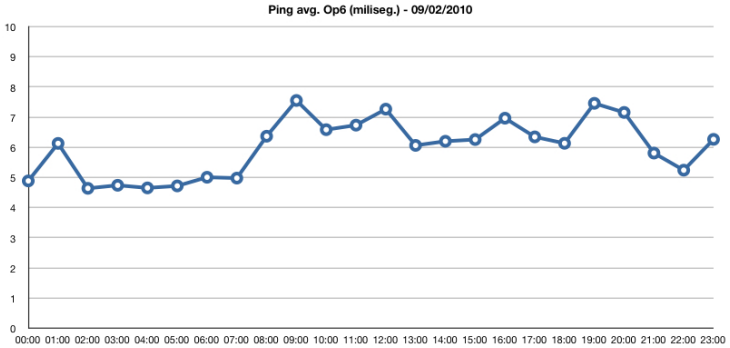


Figura 5.20: Gráfica del valor medio obtenido por el *ping* en el enlace OP6.

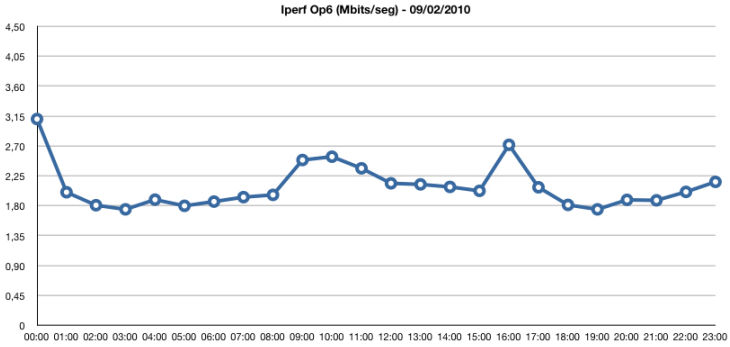


Figura 5.21: Resultado del *Iperf* en el enlace OP6.

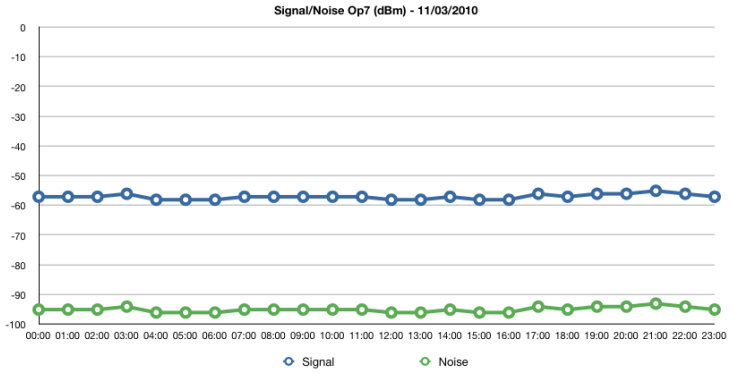


Figura 5.22: Nivel de señal y de ruido en dBm del enlace OP7.

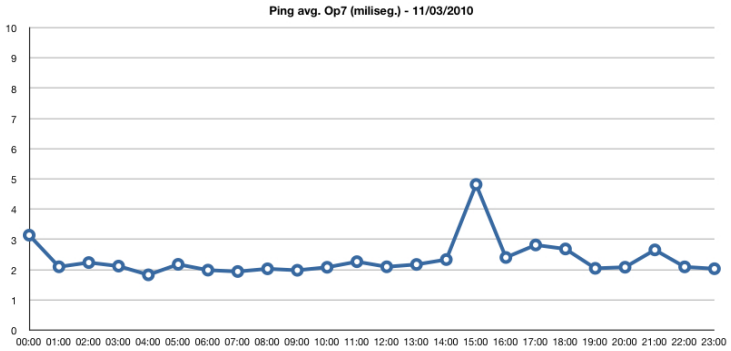


Figura 5.23: Gráfica del valor medio obtenido por el *ping* en el enlace OP7.

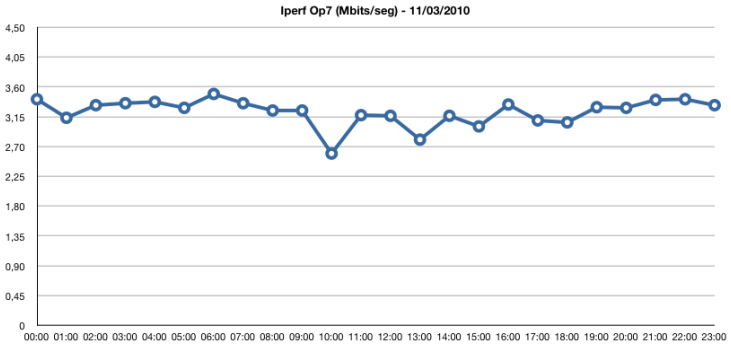


Figura 5.24: Resultado del *Iperf* en el enlace OP7.

5.5. Conclusiones y observaciones de las pruebas en el campus universitario

Antes de hablar de conclusiones, destacar que el escenario de un campus universitario, para este tipo de pruebas, muchas veces comporta muchos problemas de interferencias ya que normalmente en las universidades coexisten muchas redes.

Al principio de las pruebas, la computadora embebida que se asignó al edificio *McGregor* era la *Pronghorn Metro SBC* pero en las pruebas de laboratorio esta computadora embebida generó errores. Inicialmente se probó con cuatro tarjetas *Xtrem Range 2* (son de alta potencia), y después también se probó con la *Super Range 2*. Con cualquier tipo de tarjeta, la computadora embebida con el sistema operativo OpenWRT y el controlador *madwifi* se presentó muy inestable, ya que la mayoría de veces no iniciaba correctamente el sistema. Esta combinación no es muy recomendable. Otro inconveniente de esta computadora embebida es su elevado consumo; funciona a *48 VDC*.

La computadora embebida que mejor ha respondido con el sistema operativo OpenWRT ha sido la *Alix 2C0*. Esta computadora es sencilla, comparada con las demás que se analizaron, dispone sólo de dos puertos *MiniPCI* y no puede alimentar a tarjetas de alta potencia (*Xtrem Range 2* y *Engenius EMP-8602 + S*), pero ha mostrado muy buenos resultados. No se bloqueó ninguna vez durante todo el tiempo de las pruebas, ni tan solo se ha reiniciado. Las tarjetas con las que ha sido probada han sido las *Super Range 2* y las *R52H*.

Las computadoras embebidas *Avila GW2348-4* y *Pronghorn SBC 250* funcionando con OpenWRT han tenido un comportamiento similar. Son computadoras muy distintas, la *Pronghorn SBC 250* funciona a *5 VDC*, solo dispone de dos puertos *MiniPCI* y no pude alimentar a tarjetas de alta potencia, la *Avila GW2348-4* funciona a *12 VDC* y tiene cuatro puertos *MiniPCI*. Las dos computadoras han funcionado muy bien y han obtenido también muy buenos resultados. Si han captado algún valor fuera de lo esperado ha sido por inconvenientes en la red y no por inestabilidades en su sistema. El OpenWRT y el controlador*madwifi* han funcionado muy bien con estas dos computadoras embebidas.

Capítulo 6: Prueba de largo alcance en campo

Esta prueba de largo alcance se hace en un escenario rural real de comunicaciones. Es una prueba para ver el funcionamiento real de los sistemas de telecomunicación, su respuesta y las posibilidades de solución que ofrece a las necesidades planteadas. Es la prueba con los resultados más significativos, ya que se lleva a cabo en el mismo lugar donde serán usados los sistemas.

En este caso, el escenario escogido es la red que gestiona el GTR en la zona del río Napo en Iquitos (Perú)[1]. La red está funcionando desde *Mazán* hasta *Cabo Pantoja* (cerca de Ecuador) a lo largo del Napo peruano, pero el enlace escogido para la prueba es desde *Mazán* hasta *Huaman Urco*.

El enlace se llevara a cabo exactamente entre las comunidades de *Mazán* (Latitud: 3º29'59.92"S - Longitud: 73º5'28.01"O) y *Huaman Urco* (Latitud: 3º19'7.61"S - Longitud: 73º13'1.91"O) con una distancia de separación de 24.6 kilómetros en línea recta. A continuación se ve una imagen del enlace captada desde satélite mediante *GoogleEarth*:

Figura 6.1: Imagen de GoogleEarth del enlace de la red situada en el río Napo.

El escenario se caracteriza por ser una zona de selva accesible solo por vía fluvial,

[1] http://commons.wikimedia.org/wiki/File:Red_Napo_GTR-PUCP.JPG

poblada de una vegetación muy densa y alta que solo se podrá obviar mediante torres altas (alrededor de 70 metros). Cabe destacar que la red ya hace tiempo que está funcionando, aunque con otro tipo de sistema de telecomunicación. Con ello, la infraestructura como las torres, la energía (mediante paneles solares y baterías) o las antenas (bien direccionadas hacia los otros puntos de la red), ya está instalada y funcionando bien, cosa que facilita mucho el llevar a cabo las pruebas.

A continuación se empieza a analizar las características del enlace y su configuración.

6.1. Características y diseño del enlace

Este tipo de pruebas sería conveniente de hacerlas con todas las computadoras embebidas y tarjetas posibles pero, por razones de costo y logísticas del grupo, es inviable. Por ello se ha escogido el uso de las computadoras embebidas *Alix 2C0* **(ver sección 2.4.1)** combinadas con las tarjetas *Super Range 2 (SR2)* **(ver sección 2.6.1)**. La computadora embebida para la prueba se ha escogido en base a su consumo de *12 VDC* **(ver sección 4.1.3)** y a su fiabilidad, ya que el GTR hace tiempo que trabaja con este *hardware*.

Como en pruebas anteriores la configuración de este enlace se hará con el modelo *AP* (punto de acceso) y *STA* (cliente). A continuación se puede ver la configuración:

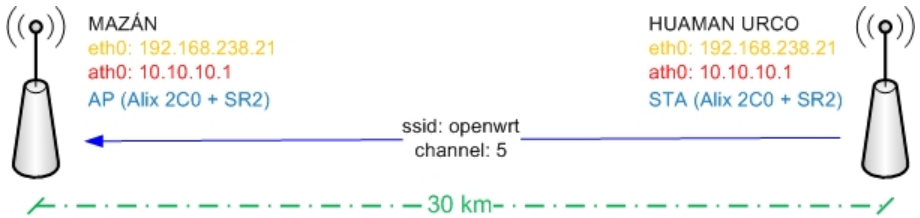


Figura 6.2: Configuración del enlace de pruebas en la red del río Napo.

Los archivos de configuración son los mismos que en las pruebas anteriores, los archivos *network* y *wireless* y el *script redes* que modifica la configuración al iniciar del equipo. Todos los archivos se encuentran adjuntos en el anexo **A.8**.

Las pruebas están orientadas a recopilar información de rendimiento (con *iperf*), a conocer el nivel de señal y el nivel de ruido (con *iwconfig*) y en conocer la *latencia* de *ping* (*max*, *min* y *avg*).

6.2. Resultados de las pruebas en largo alcance

La prueba de rendimiento, como en las pruebas hechas en el campus universitario, se hace mediante *iperf*. Las siguientes imágenes muestran los resultados obtenidos:

```
root@OpenWrt:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.10.10.2 port 5001 connected with 10.10.10.1 port
47088
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-60.1 sec    103 MBytes  14.3 Mbits/sec
```

Figura 6.3: *Iperf* de 60 segundos, del AP hacia el STA, con un *Bit Rate* de 36 Mbits/s.

```
root@OpenWrt:~# iperf -s
------------------------------------------------------------
---
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
---
[  4] local 10.10.10.1 port 5001 connected with 10.10.10.
2 port 50918
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-60.1 sec    130 MBytes  18.1 Mbits/sec
```

Figura 6.4: *Iperf* de 60 segundos, del STA hacia el AP, con un *Bit Rate* de 36 Mbits/s.

A continuación se puede ver dos imágenes de *iperf* en direccion de STA hacia AP y evaluado en 18 Mbits/s y en 54 Mbits/s:

```
root@OpenWrt:~# iperf -s
------------------------------------------------------------
---
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
---
[  4] local 10.10.10.1 port 5001 connected with 10.10.10.
2 port 50436
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-60.1 sec  74.6 MBytes  10.4 Mbits/sec
```

Figura 6.5: *Iperf* de 60 segundos, del STA hacia el AP, con un *Bit Rate* de 18 Mbits/s.

```
root@OpenWrt:~# iperf -s
------------------------------------------------------------
---
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
---
[  4] local 10.10.10.1 port 5001 connected with 10.10.10.
2 port 51040
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-60.1 sec    132 MBytes  18.4 Mbits/sec
```

Figura 6.6: *Iperf* de 60 segundos, del STA hacia el AP, con un *Bit Rate* de 54 Mbits/s.

Con los resultados anteriores se puede comprobar que a partir de *36 Mbits/s* ya no se mejora significativamente el rendimiento del enlace.

A continuación se adjuntan dos imágenes de *iwconfig*, ejecutado en el *AP* y en el *STA*. En ellas podemos ver las diferentes potencias del enlace:

```
root@OpenWrt:~# iwconfig
lo        no wireless extensions.

wifi0     no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

br-lan    no wireless extensions.

ath0      IEEE 802.11g  ESSID:"openwrt"  Nickname:""
          Mode:Master  Frequency:2.432 GHz  Access Point: 00:0C:42:18:A2:EA
          Bit Rate:0 kb/s   Tx-Power:23 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=22/70  Signal level=-74 dBm  Noise level=-96 dBm
          Rx invalid nwid:23682  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Figura 6.7: *Iwconfig* ejecutado en el AP para ver las características del enlace.

```
root@OpenWrt:~# iwconfig
lo        no wireless extensions.

wifi0     no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

br-lan    no wireless extensions.

ath0      IEEE 802.11g  ESSID:"openwrt"  Nickname:""
          Mode:Managed  Frequency:2.432 GHz  Access Point: 00:0C:42:18:A2:EA
          Bit Rate:36 Mb/s   Tx-Power:16 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=23/70  Signal level=-73 dBm  Noise level=-96 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Figura 6.8: *Iwconfig* ejecutado en el STA para ver las características del enlace.

Se puede ver que los resultados tomados en el *AP* o en el *STA* no varían mucho. El nivel de señal no es muy bueno pero está dentro del rango de valores aceptados para una buena transmisión (mayor a *-80 dBm*).

La siguiente prueba consiste en hacer una cantidad de *pings* (en este caso del *AP* al *STA* pero al revés daría valores muy parecidos), para conocer la *latencia* del canal, el valor máximo, el mínimo y la media. Las siguientes imágenes muestran los valores obtenidos:

```
--- 10.10.10.2 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max = 0.738/1.023/2.683 ms
```

Figura 6.9: *Ping* de 100 paquetes para conocer la *latencia* (*min*, *max* y *avg*).

```
--- 10.10.10.2 ping statistics ---
20 packets transmitted, 20 packets received, 0% packet loss
round-trip min/avg/max = 1.185/1.414/2.165 ms
```

Figura 6.10: *Ping* de 1000 paquetes para conocer la *latencia* (*min*, *max* y *avg*).

Hacer *ping* con 1000 paquetes, al trabajar con más valores y hacer la prueba durante más tiempo, proporciona un resultado de latencia más fiel al valor real.

6.3. Conclusiones y observaciones de las pruebas en largo alcance

Esta es una de las pruebas más significativas ya que se esta trabajando con un enlace real que ya esta trabajando. Por consiguiente, los valores que se obtienen son valores totalmente reales a nivel de errores, desviaciones, etc.. Este tipo de pruebas son muy necesarias si se quiere hacer un enlace real y en éste caso hubiera sido bueno poder hacer las pruebas con todas las computadoras embebidas y tarjetas posibles, pero por el costo de tiempo y de dinero que ello implica se descarto esta opción y se redujo la prueba con una única computadora embebida y una tarjeta. Los resultados obtenidos, en general han sido muy buenos.

Cabe decir que los días que se hicieron las pruebas estuvo nublado e incluso llovió. La siguiente imagen muestra la antena de *Huaman Urco* y también se puede distinguir el cielo nublado:

Figura 6.11: Día lluvioso de la prueba de largo alcance en el Napo.

La lluvia no afecta mucho a los valores pero efectivamente no es lo mismo que un día totalmente soleado. La zona del río *Napo* es una zona donde el clima es muy variante y a menudo llueve y hay tormentas eléctricas. Es por ello que es muy importante que todas las torres tengan conexión a *tierra*.

Otro punto muy importante es la buena orientación de las antenas. Esta tarea se pudo ahorrar porque el enlace ya esta en funcionamiento, pero es una de las tareas principales en tanto que los resultados de una conexión pueden variar mucho con una buena o mala orientación.

Para esta prueba se hubo de tener en cuenta las restricciones de nivel de potencia y de las antenas impuestas por la ley peruana **[10]**.

Capítulo 7: Conclusiones generales y observaciones

A modo de conclusión general decir que para escoger un sistema de telecomunicación que cumpla con garantías las necesidades planteadas se deben de hacer muchas pruebas y se deben de tener en cuenta varios puntos. Primero se debe de tener en cuenta las necesidades que se tienen que cubrir y el presupuesto del que se dispone; Las principales pruebas que se tienen que hacer son:

Se deben hacer pruebas de consumo de los sistemas. El consumo es muy importante conocerlo ya que las computadoras embebidas se alimentarán con sistemas fotovoltáicos formados por paneles y baterías. Como se ha visto, el consumo medio de las computadoras embebidas trabajando es de unos *9 W*, valor totalmente sostenible por los sistemas fotovoltáicos. En el caso de la computadora embebida *Pronghorn Metro SBC* el consumo es más elevado; funciona a *48 VDC* y puede trabajar con cuatro tarjetas *MiniPCI* de alta potencia.

Otra prueba importante es la prueba de rendimiento. Las pruebas hechas en campo nos demuestran que con el OpenWRT, las tarjetas *Super Range 2*, y con las computadora embebidas *Alix 2C0*, se puede alcanzar mas de *18 Mbps* de velocidad de transmisión, en un enlace de 30 kilómetros. Con ello (la velocidad de transmisión va decrementando según los saltos que tiene un red) se podría crear una red de hasta unos 20 enlaces, garantizando un mínimo de *2 Mbps* de velocidad de transmisión.

La combinación que mejor ha respondido a las pruebas ha sido la computadora embebida *Alix 2C0* con el sistema operativo OpenWRT. La computadora es sencilla, comparada con las demás que se analizaron, dispone sólo de dos puertos *MiniPCI* y no puede alimentar a tarjetas de alta potencia (*Xtrem Range 2* y *Engenius EMP-8602 + S*), pero ha mostrado muy buenos resultados. No se bloqueó ninguna vez durante todo el tiempo de las pruebas, ni tan solo se reinició.

Las computadoras embebidas *Avila GW2348-4* y *Pronghorn SBC 250* también funcionaron muy bien con el sistema operativo OpenWRT y el controlador *madwifi*, y se obtuvieron muy buenos resultados.

La computadora embebida *Pronghorn Metro SBC* con cualquier tipo de tarjeta, con el sistema operativo OpenWRT y el controlador *madwifi* se presenta muy inestable. La mayoría de veces no inicia correctamente el sistema. Esta combinación no se recomienda.

Para garantizar la robustez de un sistema operativo, como en este caso el OpenWRT, también es importante hacer pruebas que impliquen una largo periodo de tiempo (cuanto más mejor) para descartar posibles fallas por bloqueo del sistema. Cabe recordar que computadoras embebidas como la *Pronghorn Metro SBC* nos han generado este tipo de errores. Por otro lado, computadoras embebidas como la *Alix 2C0* han funcionado siempre sin bloquearse.

Así pues, es muy importante hacer varios tipos de pruebas para garantizar un buen funcionamiento de las combinaciones entre computadoras embebidas con las tarjetas, el software y las antenas.

Una observación importante es el uso de sistemas OEM (Original Equipment Manufacturer). Si el uso de estos sistemas garantiza las necesidades planteadas entonces son una solución muy buena, ya que son sistemas robustos, de costo muy bajo comparado con los sistemas comerciales y con un rendimiento aceptable en larga distancia.

Apéndice A

Anexos

A.1. Datasheets computadoras embebidas

PC Engines™

alix2c0	System board
Status	Superseded by alix2d0.
Part numbers	alix2c0 = 2 LAN / 2 miniPCI / LX700 / 128 MB
Spec	CPU: 433 MHz AMD Geode LX700 DRAM: 128 MB DDR DRAM Storage: CompactFlash socket Power: DC jack or passive POE, min. 7V to max. 20V Three front panel LEDs, pushbutton Expansion: 2 miniPCI slots, LPC bus Connectivity: 2 Ethernet channels (Via VT6105M 10/100) I/O: DB9 serial port Board size: 6 x 6" (152.4 x 152.4 mm) - same as WRAP.1E Firmware: tinyBIOS
Customer options	44 pin IDE, I2C bus, buzzer, RTC battery
Manufacturer	PC Engines
Origin	Taiwan
Support info	ALIX.2 series
Schematic	PDF

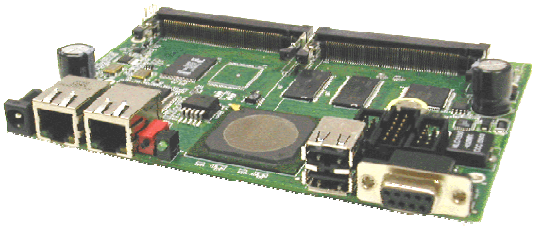
Avila GW2348-4 Network Computer

A full featured quad Mini-PCI Intel® XScale® IXP425/533MHz processor for enterprise and residential network applications

The GW2348-4 is a member of the Gateworks Avila Network Processor family. The GW2348-4 meets the requirements for enterprise and residential network applications. This network processor consists of an Intel® IXP425 XScale® operating at 533MHz, 64Mbytes of SDRAM, and 16Mbytes of Flash. Peripherals include four Type III Mini-PCI sockets, two 10/100 Base-TX Ethernet ports with IEC-6100-4 ESD and EFT protection, and Compact Flash socket. Additional features include digital I/O, serial EEPROM, real time clock with battery backup, system monitor to track operating temperature and input, two RS-232 serial ports for management and debug, fan controller, and watchdog timer. The GW2348 also supports USB as an ordering option. Power is applied through a dedicated power connector or through any Ethernet connector with the unused signal pairs in a passive power over Ethernet architecture. The mechanical dimensions and front panel connector locations are GW2358 compatible. A board support package is included for Linux 2.6 operating systems.

FEATURES

- **Intel® XScale® IXP425 533MHz Processor**
- **64Mbytes SDRAM Memory**
- **16Mbytes Flash Memory**
- **Compact Flash Socket**
- **Four Type III Mini-PCI Sockets**
- **Two 10/100 Base-TX Ethernet Ports**
- **Two RS-232 Serial Ports**
- **General Purpose Digital I/O**
- **1Kbyte Serial EEPROM**
- **Battery Powered Real Time Clock**
- **Voltage and Temperature Monitor**
- **Thermally Activated Fan Controller**
- **Watchdog Timer**
- **Front Panel LED and Reset Switch**
- **Passive Power Over Ethernet**
- **Reverse Voltage and Transient Protection**
- **9 to 48VDC Input Voltage Range**
- **20W Shared Between Mini-PCI Sockets**
- **5W Typical Operating Power**
- **-40°C to 85°C Operating Temperature**
- **Linux v2.6 Board Support Package**
- **1 Year Warranty**
- **Optional Dual Type A USB Host Ports**



SPECIFICATIONS

ELECTRICAL

Input Voltage
- 9 to 48VDC

Operating Current
- 0.2A Typical @ 24VDC

MECHANICAL

Dimensions
- 4.0in x 6.0in x 1.2in (102mm x 152mm x 30mm)

Weight
- 5 oz (142g)

ENVIRONMENTAL

Operating Parameters
- Temperature: -40°C to +85°C
- Humidity (non-condensing): 20% to 90%
- MTBF: 60 years @ 55°C

Storage Parameters
- Temperature: -40°C to +85°C
- Humidity (non-condensing): 5% to 95%

ADI ENGINEERING

Product Brief

Pronghorn SBC™ Single and Dual Radio Wireless Router Board

Pronghorn SBC™ is ADI Engineering's low-cost, high-performance, standards-compliant dual radio wireless router board based on the Intel IXP42x family of network processors. Pronghorn SBC is targeted at single and dual radio applications for WISPs, OEMs and system integrators in wireless mesh, indoor and outdoor access points, hotspots, and wireless CPE devices.

Designed to fill the void created by the discontinuance of popular Geode-based wireless router boards, Pronghorn SBC™ is a highly cost effective replacement that also delivers significant throughput increases, enhanced standards compliance, and higher power radio compatibility.

ADI offers three versions of Pronghorn SBC™, optimized for a variety of different applications including core mesh, mesh repeater, low-cost CPE, and AP and hotspot controllers.

As an Associate Member of the Intel Communications Alliance, ADI leads the market supporting XScale with reference designs, custom designs, manufacturing, and wireless router boards.

ADI's experience and commitment show in all our WISP products.

Pronghorn SBC Key Features

- **Intel IXP42x Network Processor Technology** – The IXP42x family offers up to 10x the performance of common Geode SC1100-based wireless router boards, for significant increases in throughput and capacity.
- **High-Power Radio Support** – Pronghorn SBC™ provides 6.5W of continuous MiniPCI power per slot, for complete compatibility with the latest high-power radios.
- **802.3af Compliant PoE** – Pronghorn SBC™ offers 802.3af compliant PoE, for maximal interoperability and compliance with key electrical safety and networking standards that cannot be met with passive PoE designs of competitive products.
- **Built-In Ethernet Surge Protection** – Pronghorn SBC™ 210 and 250 offer built-in surge protection on the primary Ethernet port meeting the requirements of GR1089.
- **Off-the-Shelf Indoor and Outdoor Enclosures** – Pronghorn SBC™ fits in readily available outdoor NEMA rated cast aluminum or indoor plastic cases (with optional LED lightpipes).
- **FCC Certified** – ADI provides FCC Part 15 unintentional and intentional radiator certifications. Using our RM1 MiniPCI radio, "do it yourself" systems are fully FCC compliant.
- **Spread Spectrum Clocking of CPU, SDRAM, PCI** – ADI boards are alone in the industry in their use of spread spectrum clocking, which reduces peak noise emissions up to 10dB compared to competitive products.
- **Broad Software Support** – MadWiFi, Linux 2.6, RoamAD, Antcor IkarusOS, WiliBox, Devicescape, OpenWRT, RedBoot
- **RoHS Compliant, Lead-Free Design**

ADI ENGINEERING

Product Brief

Pronghorn Metro SBC™ Quad-Radio Wireless Router Board

Pronghorn Metro SBC™ is ADI Engineering's high-performance, standards-compliant, extended temperature quad radio wireless router board based on the Intel IXP425 network processor. Pronghorn Metro SBC™ is targeted at multi-radio applications for WISPs, OEMs and system integrators in wireless mesh, outdoor access points, hotspots, long-range point-to-point links and other applications.

Pronghorn Metro SBC™ is a carrier grade router board delivering higher performance and trouble-free operation in demanding outdoor applications. High-power, electrical isolation, surge protection, FCC certification, and spread spectrum clocking are key features unique to Pronghorn Metro SBC™. Yet Pronghorn Metro SBC™ is very competitively priced.

As an Associate Member of the Intel Communications Alliance, ADI leads the market supporting XScale with reference designs, custom designs, manufacturing, and wireless router boards.

ADI's experience and commitment show in all our WISP products.

Pronghorn Metro SBC™ Key Features

- **Intel IXP425 Network Processor Technology** – The 533 MHz IXP425 offers up to 10x the performance of SC1100-based router boards, for significant increases in throughput.
- **High-Power Radio Support** – Pronghorn Metro SBC™ provides 25W total of continuous power to its four MiniPCI slots as required by the latest high-power radios, including Ubiquiti XR series and Wavesat WiMAX.
- **Electrically Isolated High-Power PoE** – Pronghorn Metro SBC™ offers 1500VRMS isolation on its high-power PoE input, along with inrush and overcurrent protection, for unmatched robustness and standards compliance.
- **Built-In Ethernet Surge Protection** – Pronghorn Metro SBC™ offers built-in Ethernet surge protection meeting GR1089 requirements.
- **Off-the-Shelf Enclosure Support** – ADI's AE1 cast NEMA rated cast aluminum case is a cost-effective, rugged and flexible outdoor enclosure for Pronghorn Metro SBC™.
- **FCC System Certification** – Using our RM1 MiniPCI radio, "do it yourself" systems built with Pronghorn Metro SBC™ are fully FCC compliant.
- **Spread Spectrum Clocking of CPU, SDRAM, PCI** – ADI boards are alone in the industry in their use of spread spectrum clocking, which reduces peak noise emissions up to 10dB compared to competitive products.
- **Broad Software Support** – MadWiFi, RoamAD, Antcor IkarusOS, StarOS, WiliBox, Devicescape, OpenWRT, Linux 2.6, RedBoot
- **RoHS compliant, lead-free design**

A.2. Datasheets tarjetas

UBIQUITI NETWORKS

SUPER RANGE 2

Powerful Range and Throughput Performance for Wi-Fi Networks

CARD INFORMATION	
Chipset	Atheros, 4th Generation, AR5213
Radio Operation	IEEE 802.11b/g, 2.4GHz
Interface	32-bit mini-PCI Type IIIA
Operation Voltage	3.3VDC
Antenna Ports	u.fl (main), MMCX (secondary)
Temperature Range	-40C to +80C
Security	WPA, WPA2, AES-CCM & TKIP Encryption, 802.1x, 64/128/152bit WEP
Data Rates	6Mbps, 9Mbps, 12Mbps, 24Mbps, 36Mbps, 48Mbps. 54Mbps
TX Channel Width Support	5MHz / 10MHz / 20MHz / 40MHz
RoHS Compliance	YES

REGULATORY INFORMATION	
Wireless Modular Approvals	FCC Part 15.247, CE(100mW limited)

RADIO OPERATING FREQUENCY 2412-2462 MHz (2312-2732 MHz*)

TX SPECIFICATIONS				RX SPECIFICATIONS			
	DataRate	TX Power	Tolerance		DataRate	Sensitivity	Tolerance
802.11b	1Mbps	26 dBm	+/-1dB	802.11b	1Mbps	-97 dBm	+/-1dB
	2Mbps	26 dBm	+/-1dB		2Mbps	-96 dBm	+/-1dB
	5.5Mbps	26 dBm	+/-1dB		5.5Mbps	-95 dBm	+/-1dB
	11Mbps	26 dBm	+/-1dB		11Mbps	-92 dBm	+/-1dB
802.11g OFDM	6Mbps	26 dBm	+/-1dB	802.11g OFDM	6Mbps	-94 dBm	+/-1dB
	9Mbps	26 dBm	+/-1dB		9Mbps	-93 dBm	+/-1dB
	12Mbps	26 dBm	+/-1dB		12Mbps	-91 dBm	+/-1dB
	18Mbps	26 dBm	+/-1dB		18Mbps	-90 dBm	+/-1dB
	24Mbps	26 dBm	+/-1dB		24Mbps	-86 dBm	+/-1dB
	36Mbps	24 dBm	+/-1dB		36Mbps	-83 dBm	+/-1dB
	48Mbps	22 dBm	+/-1dB		48Mbps	-77 dBm	+/-1dB
	54Mbps	21 dBm	+/-1dB		54Mbps	-74 dBm	+/-1dB

ADJUSTABLE CHANNEL SIZE SUPPORT (Increase Channel Capacity or Increase Throughput)			
5MHz	10MHz	20MHz	40MHz (Turbo)

CURRENT CONSUMPTION INFORMATION

TX CURRENT CONSUMPTION				RX CURRENT CONSUMPTION			
	DataRate	Current	Tolerance		DataRate	Sensitivity	Tolerance
802.11b	1Mbps	1.10 A	+/-100mA	802.11b	1Mbps	350 mA	+/-100mA
	2Mbps	1.10 A	+/-100mA		2Mbps	350 mA	+/-100mA
	5.5Mbps	1.10 A	+/-100mA		5.5Mbps	350 mA	+/-100mA
	11Mbps	1.10 A	+/-100mA		11Mbps	350 mA	+/-100mA
802.11g OFDM	6Mbps	1.10 A	+/-100mA	802.11g OFDM	6Mbps	350 mA	+/-100mA
	9Mbps	1.10 A	+/-100mA		9Mbps	350 mA	+/-100mA
	12Mbps	1.10 A	+/-100mA		12Mbps	350 mA	+/-100mA
	18Mbps	1.10 A	+/-100mA		18Mbps	350 mA	+/-100mA
	24Mbps	1.10 A	+/-100mA		24Mbps	350 mA	+/-100mA
	36Mbps	1.00 A	+/-100mA		36Mbps	350 mA	+/-100mA
	48Mbps	0.90 A	+/-100mA		48Mbps	350 mA	+/-100mA
	54Mbps	0.80 A	+/-100mA		54Mbps	350 mA	+/-100mA

RANGE PERFORMANCE	
Indoor (Antenna Dependent):	Up to 200meters
Outdoor (Antenna Dependent):	Over 50km

DRIVER INFORMATION	
Operating System Support	Linux MADWIFI, WindowsXP, Windows2000
Advanced Mobility / QuickHandoff	WindowsXP/2000 Utility with Enhanced Mobility Driver from Ubiquiti
Cisco Support	CCX 4.0 Supported Driver/Utility also available from Ubiquiti

For help with MADWIFI or other Special Driver Support, Please e-mail support@ubnt.com

495-499 Montague Expwy. Milpitas, CA 95035
San Jose, CA 95112
T (408)-942-3085 F (408)-351-4973
http://www.ubnt.com

routerBOARD.com

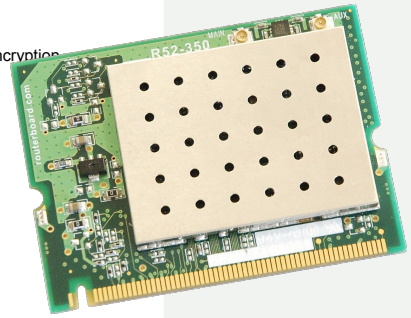
2.4/5GHz 802.11a+b+g High Power Wireless Mini-PCI Card (R52H)

- **Turbo, 802.11a, 802.11b and 802.11g IN ONE**
- **Operates in both 2.4 GHz and 5 GHz wireless bands**
- **Support MikroTik Nstreme**
- **Extended distances and higher speeds due to better output signal power**
- **FCC and CE approval**

MikroTik introduces the R52H wireless 802.11a+b+g miniPCI card for multiband high speed applications, with up to 350mW output power. It works on 2.192-2.539 and 4.920-6.100GHz frequency range and supports Turbo mode for faster transfers. The card performs best when coupled with the MikroTik RouterOS.

R52H is optimized to work with MikroTik Nstreme protocol to reach extra long distances at a great speed. The Nstreme protocol is MikroTik proprietary wireless protocol created to overcome speed and distance limitations of IEEE 802.11 standards and to extend point-to-point and point-to-multi point wireless link performance. The new Nstreme-dual protocol designed to provide real full-duplex communications over wireless with a pair of wireless cards – one for transmitting data and one for receiving.

Specifications



Chipset:	Atheros AR5414
Standards:	IEEE802.11a, IEEE802.11b, IEEE802.11g
Media Access:	CSMA/CA with ACK architecture 32-bit MAC
Security:	Hardware-based 64/128 bit WEP, TKIP and AES-CCM encryption WPA, WPA2, 802.1x
Modulation:	802.11b+g: DSSS, OFDM for data rate >30Mbps 802.11a: OFDM
Host Interface:	Mini-PCI form factor; Mini-PCI Version 1.0 type 3B suggested only for motherboards that are produced after 2004
Connectors:	Two U.fl connectors
Wi-Fi:	WECA Compliant
Certifications:	FCC, EC
Powering:	3.3V +/- 10% DC; 800mA max (600mA typ.)
Frequencies:	
802.11b/g	2.192 – 2.507 (5 MHz step); 2.224 – 2.539 (5MHz step)
802.11a	4.920 – 6.100 (5 MHz step)
Transfer Data Rate:	802.11b:11,5.5,2,1 Mbps, auto-fallback 802.11g(Normal mode):54,48,36,24,18,12,9,6 Mbps, auto-fallback 802.11g(Turbo mode):108,96,72,48,36,24,18,12 Mbps, auto-fallback 802.11a(Normal mode):54,48,36,24,18,12,9,6 Mbps, auto-fallback 802.11a(Turbo mode):108,96,72,48,36,24,18,12 Mbps, auto-fallback
Output Power / Receive Sensitivity*:	
IEEE 802.11a:	24dBm / -90dBm @ 6Mbps 19dBm / -70dBm @ 54Mbps
IEEE 802.11b:	25dBm / -92dBm @ 1Mbps 25dBm / -87dBm @ 11Mbps
IEEE 802.11g:	25dBm / -90dBm @ 6Mbps 20dBm / -70dBm @ 54Mbps
	* - Tested to comply with IEEE802.11 BER specs with RouterOS v2.9 and newer only.
Supported OS:	MikroTik RouterOS, Windows XP, GNU/Linux
Operation Temp.:	-20°C to 70°C
Storage Temp.:	-65°C to 100°C
Humidity range:	Operating 5% to 95% (non-condensing)
Dimensions:	6.0cm x 4.5 cm
Weight:	20 grams

Warning: it is always advised to keep an antenna connected during high power tx to avoid damage to the amplifier circuit.

Contact support@routerboard.com for support questions.

Mikrotikls SIA, Pernavas 46, LV-1009, Riga, LATVIA; Phone: +371 7317700; Fax: +371 7317701; E-mail: sales@routerboard.com
http://www.routerboard.com RouterBOARD, RouterOS and MikroTik are trademarks of Mikrotikls SIA

UBIQUITI NETWORKS

XTREMERange2

Carrier-Class 2.4GHz 802.11b/g Radio Module

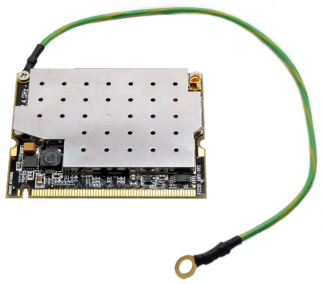
The XtremeRange series of radio modules by Ubiquiti leverages our strong knowledge and experience gained from customer interactions, field performance evaluations, and lab research; and improves upon the original and highly successful SuperRange series of high-performance 802.11 radio cards. The XtremeRange2 represents the first true carrier-class 802.11b/g-based 2.4GHz radio module specifically designed for mesh, bridging, and infrastructure applications requiring the highest levels of performance and reliability without compromise.

Designed to Link Farther and Faster

Built to Last Outdoors in Harshest Environments

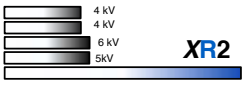
FEATURES

- 600mW Output Power
- ESD/EMP Protection
- Industry-Best Sensitivity
- Extended Temperature
- Enhanced Filtering
- 5/10/20/40 MHz Channels
- MMCX Ant. Connector



RF ESD/EMP Earth Ground Cable

RF ESD/EMP Immunity Threshold vs. Standard Cards

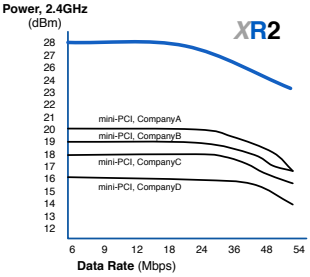


**must use included grounding cable*

Built-in HeatSink for Temperature Performance

Built for Industrial / Rugged Applications

Power, 2.4GHz (dBm)



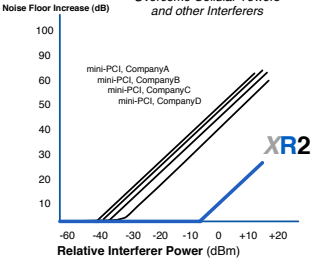
Tested with and Optimized for

MikroTik — MADWiFi

Mikrotik is a trademark of Mikrotikls SIA, Latvia

1.9GHz Rejection
Noise Floor Increase (dB)

Overcome Cellular Towers and other Interferers



802.11a/b/g
Wireless Mini-PCI Adapter
EMP-8602S

The EMP-8602S is a mini-PCI type III B High-Power card supporting dual-band (2.4GHz & 5GHz) radio operation. With the highest output power available in the market today in a very small form factor, EnGenius Technologies EMP-8602S is the perfect solution using 802.11a/b/g networking technology.

Specifications

Features:

- Output Power up to 28dBm in 11b/g mode, 22dBm in 11a mode
- Up to 54Mbps data transfer rate
- Flexible designs for embedded systems or OEM project based applications
- Fully interoperable with IEEE802.11a/b/g compliant products
- Automatic data rate scaling at 54,48,36,24,18,12,11,9,6,5.5,2,1Mbps
- Updated 64/128-bit WEP engine for improved throughput
- Uses latest IEEE802.1x Client Support(EAP-TLS, EAP-TTLS supplicant)
- Advanced power management for extended battery life
- Improved indoor multipath distortion for higher link quality in an indoor environment
- Extended tuning ranges for worldwide use and DFS/TPC for international operation
- Support for 802.11e standards for Wireless Multimedia Enhancements QoS

Network:

- **Compliant:** FCC Part 15/UL
- **Media Access Protocol:**
  CSMA/CA
- **Modulation Technology:** OFDM
  802.11a/g: OFDM (64-QAM, 16-QAM, QPSK, BPSK)
  802.11b: DSSS (DBPSK, DQPSK, CCK)
- **Security:**
  64/128-bit WEP data encryption
  IEEE802.1x Client Support(EAP-TLS, EAP-TTLS supplicant)
  WPA/WPA2- Wi-Fi Protected Access(64, 128-bit WEP with TKIP/AES, Pre-Share Key)
- **Network Standards:**
  IEEE802.11a IEEE802.11b IEEE802.11g IEEE802.1x
  IEEE802.11i IEEE802.11h IEEE802.1e IEEE802.1j

Physical Specifications:

- **Form Factor:** Mini-PCI Type IIIB
- **Dimensions:** 59.60mm X 44.45mm
- **Antenna Connector:** 2x U.FL Connectors
- **Operating Voltage:** 3.3V +/- 0.15V

Environmental:

- **Temperature Range:**
  Operating: 0°C to 70°C
  Storage: -20°C to 80°C
- **Humidity (non-condensing):**
  5%~95% Typical

Radio:

- **Frequency Band:**
  802.11a:
  5.15~5.35GHz,
  5.47~5.725GHz, 5.725~5.825GHz
  802.11b/g:
  U.S., Europe and Japan product covering 2.4 to 2.484 GHz, programmable for different country regulations
- **Radio Type:**
  Direct Sequence Spread Spectrum (DSSS)

Operation Channels:
11 for North America

Receive Sensitivity (Typical):
5.15~5.82 GHz
6Mbps @ -90dBm 54Mbps @ -74dBm
- 2.412~2.472 G(IEEE802.11g)
  6Mbps @ -92dBm 54Mbps @ -76dBm
  2.412~2.472 G(IEEE802.11b)
  11Mbps @ -92dBm 1Mbps @ -96dBm
- **Current Consumption:**
  Tx Current: 1A
  Rx Current: 400mA
  Card on Current: 400mA
  Sleep Current: 100mA
- **Available Transmit Power *subject to country regulations*:**
  5.15~5.25 GHz(IEEE802.11a)
  22dBm @6 ~ 24Mbps
  20dBm @36Mbps
  18dBm @48Mbps
  17dBm @54Mbps
  5.25~5.35GHz(IEEE802.11a)
  20dBm @6 ~ 24Mbps
  18dBm @36Mbps
  16dBm @48Mbps
  15dBm @54Mbps
  5.500~5.700GHz(IEEE802.11a)
  21dBm @6 ~ 24Mbps
  19dBm @36Mbps
  17dBm @48Mbps
  16dBm @54Mbps
  5.745~5.85GHz (IEEE802.11a)
  20dBm @6 ~ 24Mbps
  18dBm @36Mbps
  16dBm @48Mbps
  15dBm @54Mbps
  2.412~2.472G(IEEE802.11g)
  27dBm @6 ~ 24Mbps
  25dBm@36Mbps
  24 dBm@48Mbps
  23dBm@54Mbps
  2.412~2.472G(IEEE802.11b)
  28dBm. @1, 2, 5.5 and 11Mbps

EnGenius™
1580 Scenic Avenue
Costa Mesa, CA 92626
Toll Free: 888.735.7888
Tel: 714.432.8668
Fax: 714.432.8667
www.engeniustech.com

*All specifications are subject to change by EnGenius without prior notice.

A.3. Datasheets antenas

L-com Global Connectivity www.L-com.com

HyperLink Wireless brand HyperGain® 2.4 GHz 11 dBi Dual Cable Spatial Diversity Flat Patch Range Extender Antenna

Model: RE11DS

Designed & Manufactured in the USA
by HyperLink Technologies

Applications and Features

Applications:

- 2.4 GHz ISM Band
- IEEE 802.11b, 802.11g Wireless LAN
- IEEE 802.11n (Pre-N, Draft-N, MIMO) Applications
- Bluetooth®
- Public Wireless Hotspot
- WiFi
- Wireless Video Systems
- Dual Diversity / Dual Antenna Radios

Features:

RoHS Compliant

- Superior performance
- Light weight
- Durable UV-stable, UL flame rated radome
- Low loss solid brass elements
- DC Short lightning protection
- Dual 3 foot coax leads
- Spatial Diversity
- RoHS Compliant
- Can be installed for either vertical or horizontal polarization
- Optional mounting brackets available

Description

Spatial Diversity Antenna

The spatial diversity antenna is designed with 2 totally separate cross-polarized antennas inside, the RE11DS is ideal for use with wireless access points, routers and pc cards that have dual antenna ports. It is ideally suited for IEEE 802.11b, 802.11g and 802.11n wireless LANs, Bluetooth®, public wireless hotspot applications and other systems operating in the 2.4GHz ISM band.

The spatial diversity feature of this WiFi antenna is useful for operating in areas susceptible to the affects of multi-path interference. By providing spatial diversity, the radio's internal circuitry can select between the two receive antennas for better wireless reception. The RE11DS spatial diversity antenna can be installed for horizontal or vertical polarization. This polarization diversity antenna can be wall or ceiling mounted, as well as mast-mounted using U-bolts.

This spatial diversity range extender antenna features an attached dual 3-foot cable terminated with the appropriate radio connectors, eliminating the need for a separate radio pigtail adapter.

L-COM, INC. 45 BEECHWOOD DRIVE NORTH ANDOVER, MA 01845
WWW.L-COM.COM E-MAIL: SALES@L-COM.COM PHONE: 1-800-343-1455 FAX: 1-978-689-9484

HyperLink Technologies — Innovative Solutions For Wireless Communications

HyperGain® 2.4 GHz 11 dBi Dual Cable Polarization Diversity Flat Patch Range Extender Antenna

Model: RE11DP

Designed & Manufactured in the USA by HyperLink Technologies

Applications and Features

Applications:

- 2.4 GHz ISM Band
- IEEE 802.11b, 802.11g Wireless LAN
- IEEE 802.11n (Pre-N, Draft-N, MIMO) Applications
- Bluetooth®
- Public Wireless Hotspot
- WiFi
- Wireless Video Systems
- Dual Diversity / Dual Antenna Radios

Features:

RoHS Compliant

- Superior performance
- Light weight
- Durable UV-stable, UL flame rated radome
- Low loss solid brass elements
- DC Short lightning protection
- Dual Diversity
- Dual 3 foot coax leads
- Polarization Diversity
- RoHS Compliant
- Can be installed for either vertical or horizontal polarization
- Optional mounting brackets available

Description

Polarization Diversity Antenna

The dual diversity antenna is designed with 2 totally separate cross-polarized antennas inside, the RE11DP is ideal for use with wireless access points, routers and pc cards that have dual antenna ports. It is ideally suited for IEEE 802.11b, 802.11g and 802.11n wireless LANs, Bluetooth®, public wireless hotspot applications and other systems operating in the 2.4GHz ISM band.

The polarization diversity feature of this WiFi antenna is useful for operating in areas susceptible to the affects of multi-path interference. By providing both spatial and cross polarization diversity, the radio's internal circuitry can select between the two receive antennas for better wireless reception. The RE11DP dual diversity antenna can be installed for horizontal or vertical polarization. This polarization diversity antenna can be wall or ceiling mounted, as well as mast-mounted using U-bolts.

This polarization diversity range extender antenna features an attached dual 3-foot cable terminated with the appropriate radio connectors, eliminating the need for a separate radio pigtail adapter.

HyperLink Technologies, INC.

e-mail: sales@hyperlinktech.com • tel: 561-995-2256 • fax: 561-995-2432
web: www.hyperlinktech.com • 1201 Clint Moore Road • Boca Raton FL 33487

HyperLink Wireless 2.3 GHz to 6.5 GHz Ultra-Wideband (UWB) 8 dBi Radome Enclosed Log Periodic Antenna

Applications and Features

Applications:

- 802.11a, 802.11b, 802.11g and 802.11n access points and routers
- 802.16 and 802.20 applications
- WiMAX Technology
- WiFi Systems
- Homeland Security and Public Safety Services: Fire, Police, Security

Features:

- Ultra-Wideband design
- Ideal for use with multiband access points and routers
- Superior performance
- Compact size, low profile and easy to mount
- 9 inch coax lead

RoHS Compliant

Model: HG2458-08LP

Description

Superior Performance

The HyperGain® HG2458-08LP is a high performance ultra wide band log periodic antenna designed to operate from 2.3 GHz to 6.5 GHz. This Ultra-Wideband design eliminates the need to purchase different antennas for each frequency. This simplifies installations since the same antenna can be used for a wide array of wireless applications where wide coverage is desired. The broadband characteristics of the antenna enables it to operate over a very wide frequency rang with consistent gain. This antenna features 8 dBi of gain and a 60 degree beam width. This antenna is ideal for use with the following applications:

◇ 2.3 GHz WCS/CDMA band
◇ 2.4 GHz 802.11a, 802.11b, 802.11g and 802.11n
◇ 2.3-2.5 GHz, 2.5-2.7 GHz, 3.4-3.5 GHz and 3.3-3.9 GHz WiMAX
◇ 2.6 GHz MMDS, 802.16, 802.20
◇ 3.5 GHz 802.16e
◇ 4.9 GHz homeland security band
◇ 5.3 GHz, 5.4 GHz and 5.8 GHz 802.11a
◇ 5.8 GHz ISM, UNII and Mesh Networks

Rugged and Weatherproof

The internal components of this antenna are enclosed within a UV-stable white fiberglass radome for all-weather operation. It is supplied with a swivel mast mount kit.

L-COM, INC. 45 BEECHWOOD DRIVE NORTH ANDOVER, MA 01845
WWW.L-COM.COM E-MAIL: SALES@L-COM.COM PHONE: 1-800-343-1455 FAX: 1-978-689-9484

HyperLink Wireless 2.4 GHz / 5.1 GHz to 5.8 GHz Multi-Band Flat Patch Wireless LAN Antenna Model: HG2458-09P

Multi-Band Operation!

Applications and Features

Applications:

- **2.4-2.5 GHz / 5.1-5.8 GHz Frequency Range:**
  - ◇ IEEE 802.11a, 802.11b, and 802.11g devices such as access points, routers and PCI Cards
  - ◇ Ideal for Multi-Band radios (802.11a/b/g)
  - ◇ 2.4 GHz ISM, 5.3 GHz and 5.8 GHz UNII/ISM band wireless systems
  - ◇ WiFi and WiMAX applications
  - ◇ 2.4 GHz and 5.8 GHz wireless video systems
  - ◇ Bluetooth® applications
  - ◇ Public wireless hotspots

Features:

- Multi-Band operation - 2400-2500 MHz and 5125-5850 MHz
- Dual element 2.4 GHz and 5 GHz internal antennas coupled to a single feed
- Light weight
- Durable UV-stable, UL flame rated radome
- Low loss solid brass elements
- Integral N-Female connector
- Can be installed for either vertical or horizontal polarization
- Includes tilt and swivel mast mount (Optional wall mounting brackets available)

Description

The HG2458-09P is a high performance multi-band directional flat patch WiFi antenna suitable for indoor and outdoor applications in the 2.4GHz (2400-2500 MHz) and 5.1-5.8 GHz (5125-5850 MHz) band. The Multi-Band design of this antenna eliminates the need to purchase different antennas for each frequency. This simplifies installations since the same antenna can be used for a wide array of wireless applications. This WiFi antenna is lightweight and features an aesthetic UV-stable, UL flame rated white plastic radome which can also be painted to match the room or building structure. It features an integral N-Female connector.

The HG2458-09P is actually two antennas in one. A dual element 2.4 GHz antenna and a dual element 5 GHz antenna coupled to a single feed. Since many dual-band 802.11a/b/g radios share a single antenna, the HG2458-09P can split these signals so that the two separate internal 2.4 GHz and 5 GHz antennas can be used to improve performance. The HG2458-09P can also receive separate 2.4 GHz and 5 GHz signals and combine them into a single feed. These features makes this antenna ideal for mixed IEEE 802.11a, 802.11b and 802.11g wireless LANs, Bluetooth®, and public wireless hotspots.

The HG2458-09P antenna is supplied with a 60 degree tilt and swivel mast mounting kit. This antenna can be wall mounted with the optional brackets available from HyperLink.

L-COM, INC. 45 BEECHWOOD DRIVE NORTH ANDOVER, MA 01845
WWW.L-COM.COM E-MAIL: SALES@L-COM.COM PHONE: 1-800-343-1455 FAX: 1-978-689-9484

HyperLink Wireless 2.4 GHz / 4.9-5.8 GHz 14 dBi 90 Degree Dual-Band Dual-Feed Sector Panel Antenna Model: HG2458-14P-090

Applications and Features

Applications:

- 802.11a, 802.11b, 802.11g and 802.11n access points and routers
- WiMAX Technology
- WiFi Systems
- Homeland Security and Public Safety Services: Fire, Police, Security
- Wireless Internet Provider "cell" sites

Features:

- **Dual band: 2.4 GHz and 4.9 GHz to 5.9 GHz**
- All weather operation
- Dual feed via (2) Integral N-Female Connectors
- All weather operation
- 10° down-tilt mast mounting bracket and hardware

RoHS Compliant

Description

The HyperGain® HG2458-14P-090 is a high performance dual-band sector panel antenna, which combines high gain with a 90° beam-width. Its dual-band design makes it suitable for applications in the 2.4GHz (2400-2500 MHz) and 5 GHz (4900-5900 MHz) band which and eliminates the need to purchase different antennas for each frequency. This simplifies installations since the same antenna can be used for a wide array of wireless applications. This antenna is ideal for use with the following applications:

◇ 2.4 GHz 802.11a, 802.11b, 802.11g and 802.11n
◇ 4.9 GHz homeland security band
◇ 5.3 GHz, 5.4 GHz and 5.8 GHz 802.11a
◇ 5.8 GHz ISM, UNII and Mesh Networks

The HG2458-14P-090 is actually two antennas in one. A 2.4 GHz antenna and a 4.9 GHz to 5.9 GHz antenna integrated into a single enclosure. Each internal antenna is fed via its own individual connector.

This dual-band sector antenna features a heavy-duty fiberglass radome for all-weather operation. The heavy-duty mounting system allows installation adjusts from 0 to 10 degrees down tilt.

This is an ideal choice for Wireless Internet Provider "cell" sites since the cell size can be easily determined by adjusting the down-tilt angle. Horizontal coverage is a full 90 degrees.

L-COM, INC. 45 BEECHWOOD DRIVE NORTH ANDOVER, MA 01845
WWW.L-COM.COM E-MAIL: SALES@L-COM.COM PHONE: 1-800-343-1455 FAX: 1-978-689-9484

HyperLink Wireless 900 MHz 15 dBi High Performance Die Cast Parabolic Grid Antenna

Model: HG915G

Applications and Features

Applications:

- 900MHz ISM Band
- Wireless LAN systems
- Wireless Video Links
- Non Line of Sight (NLOS)
- 900MHz Cellular
- Long-range Directional Applications
- Point to Point Systems
- Point to Multi-point Systems
- Wireless Bridges
- Backhaul Applications

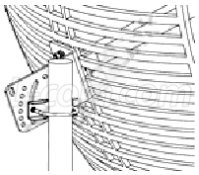
Features:

RoHS Compliant

- Superior performance
- Die Cast aluminum construction
- UV stable light gray powder coat finish
- All weather operation
- 19° beam-width
- Two piece reflector grid significantly reduces shipping costs
- Simple to assemble
- Low Wind Loading
- RoHS Compliant

Description

Superior Performance



The HyperGain® High Performance Parabolic Grid Antenna provides 15 dBi gain with a 19° beam-width for directional applications, backhaul applications, point to point systems and Non Line of Sight (NLOS) installations. It is ideally suited for 900MHz ISM and GSM bands. Typical applications include 900MHz Wireless LAN, SCADA, Wireless Video Links and 900MHz Cellular. External interference of this antenna is minimized due to the excellent front to back ratio. This antenna comes with a 30" coax lead terminated with an N-Female connector.

Rugged and Weatherproof

The HG915G construction features a rust-proof die cast aluminum reflector grid for superior strength and light weight. This antenna's 2-piece reflector grid is simple to assemble and significantly reduces shipping costs. The grid surface is UV powder coated for durability and aesthetics. The open-frame grid design minimizes wind loading.
This antenna is supplied with a 20 degree tilt and swivel mast mount kit. This allows installation at various degrees of incline for easy alignment. It can be adjusted up or down from 0° to 20°. It can mount to a 1.25" (31.8mm) to 2" (50.8mm) dia. mast.

L-COM, INC. 45 BEECHWOOD DRIVE NORTH ANDOVER, MA 01845
WWW.L-COM.COM E-MAIL: SALES@L-COM.COM PHONE: 1-800-343-1455 FAX: 1-978-689-9484

Pacific Wireless
a division of CUSHCRAFT CORPORATION

PAWGD9-18
DATA SHEET

NLOS Series Grid Dish Antenna
900 to 928 MHz Operation

*18dBi Directional Grid Dish Antenna

Features

- **High Gain Directional 900MHz Antenna**
- **Low Wind Loading Wire Grid Design**
- **Vertical Polarization**
- **18" Pigtail with Type N Female Connector Standard**

Applications

- **900 MHz ISM Band Applications**
- **900MHz Backhaul Applications**
- **Non Line of Sight Applications**
- **Point to Point Systems**

Description

The NLOS Series Grid Dish antenna system offered by Pacific Wireless is constructed of heavy duty galvanized welded steel with light gray powder coat paint overcoat for long service life. These antennas have high gain and good front to back performance to minimize external interference. They come standard with an 18" pigtail cable terminated with an N Female connector. Other connector types are available upon request.

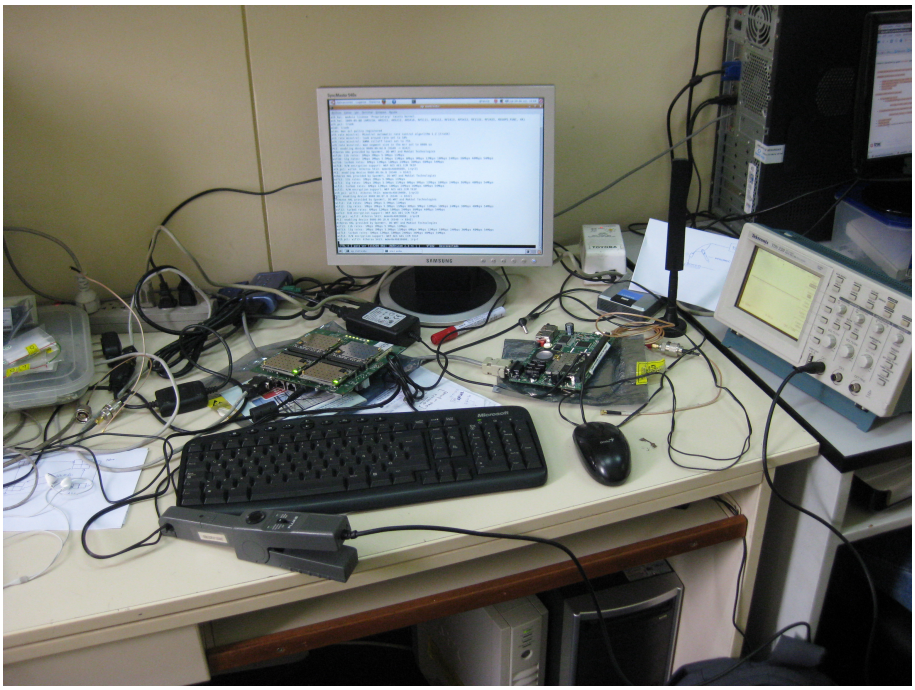
Specifications

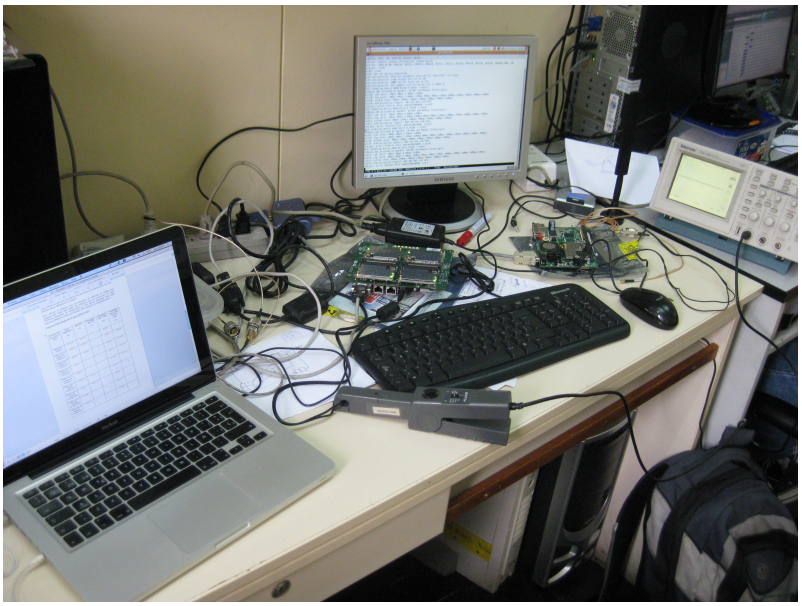
Parameter	Min	Typ	Max	Units
Frequency Range	900		928	MHz
Input Return Loss (S_{11})		-14		dB
VSWR		1.5:1		
Impedance		50		OHM
Input Power			100	W
Pole Diameter (OD)	1.5 (38)		3 (76)	Inch (mm)
Operating Temperature	-45		+70	Deg C
Rated Wind Velocity			125 (56)	Mph (m/sec)

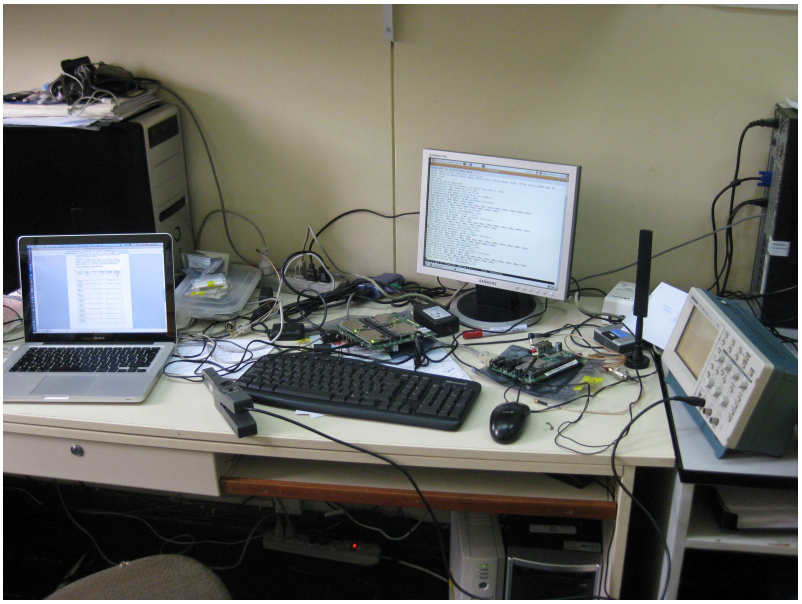
*Note: Pacific Wireless does not supply the mounting pole

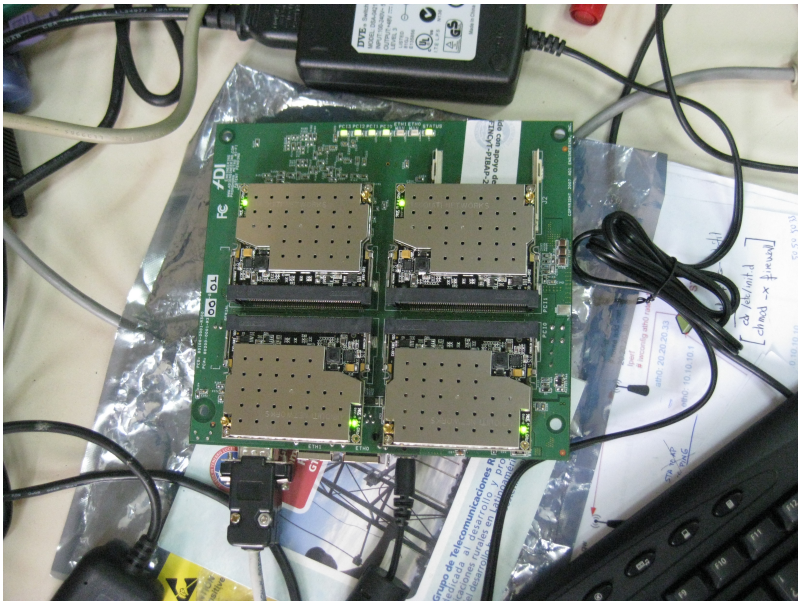
900 – 928 MHZ	GD9-18
Gain	18dBi
Beamwidth	16.5 deg
Front to Back	> 30dB
Weight	27 Lbs (10 Kg)
Dimension - Diameter	4' (1.2 m)

A.4. Fotos de la prueba de consumo de energía

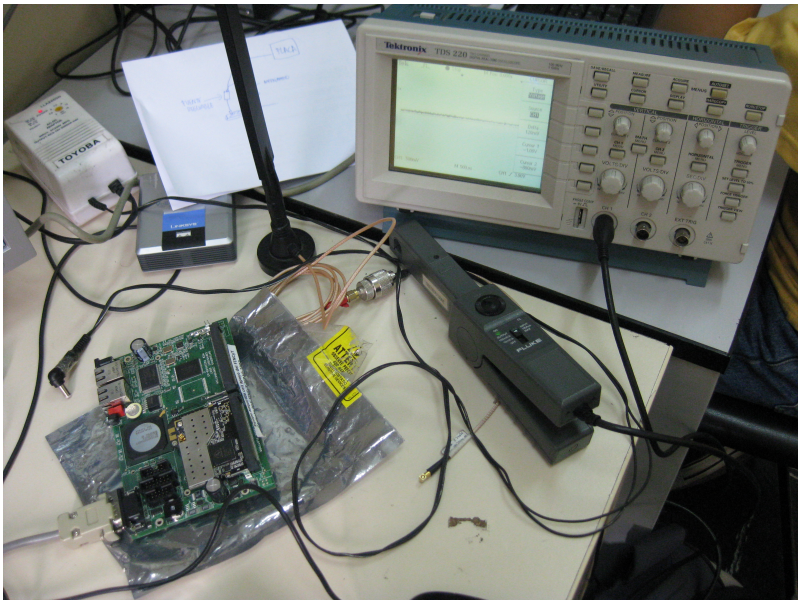


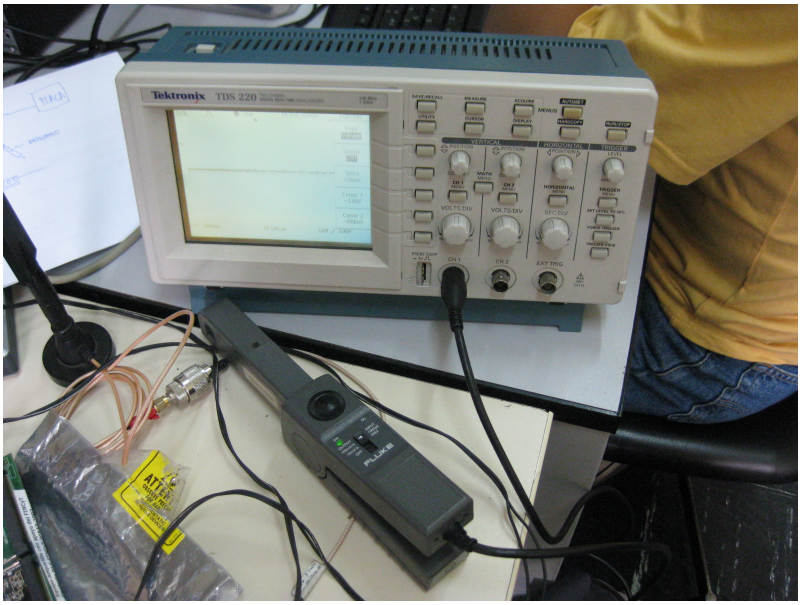




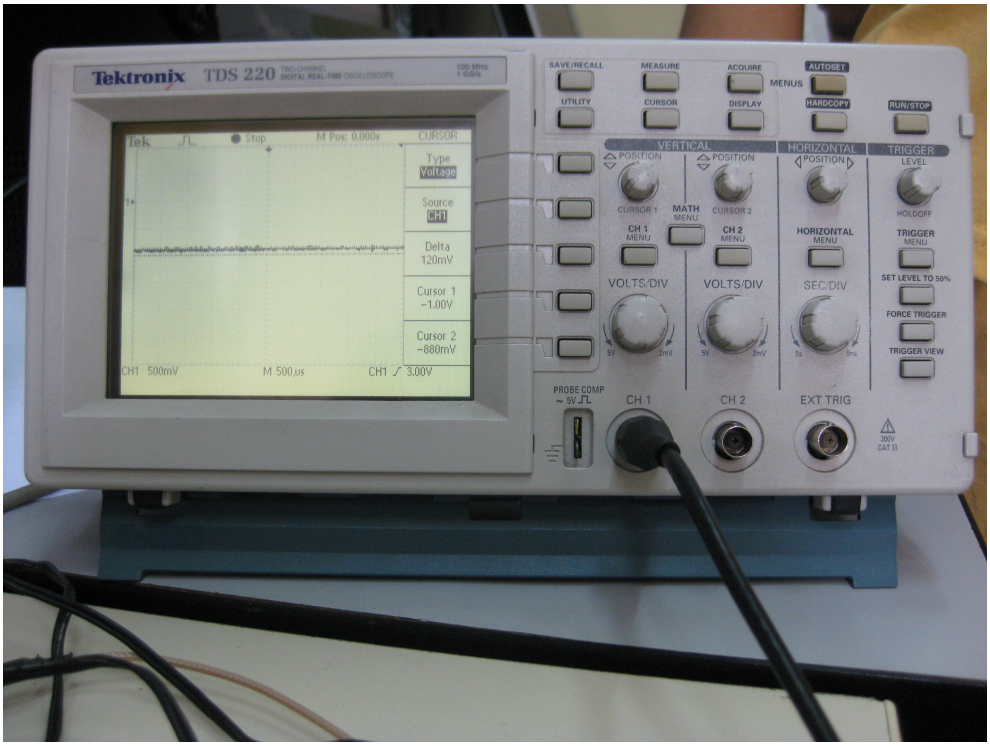



Equipo Adquirido con apoyo del FINCyT


Tektronix TDS 220
TOYOBA
FLUKE


Tektronix TDS 220
FLUKE


ON
OL
OUTPUT
INPUT
100A pk
MAX
OFF
FLUKE


Tektronix
TDS 220
SAVE/RECALL
MEASURE
ACQUIRE
AUTOSET
UTILITY
CURSOR
DISPLAY
HARDCOPY
RUN/STOP
VERTICAL
HORIZONTAL
TRIGGER
POSITION
LEVEL
MATH MENU
CH 1 MENU
CH 2 MENU
HORIZONTAL MENU
TRIGGER MENU
HOLDOFF
SET LEVEL TO 50%
FORCE TRIGGER
TRIGGER VIEW
VOLTS/DIV
SEC/DIV
PROBE COMP ~ 5V
CH 1
CH 2
EXT TRIG
Tek
Stop
M Pos: 0.000s
CURSOR
Type
Voltage
Source
CH1
Delta
120mV
Cursor 1
-1.00V
Cursor 2
-880mV
CH1 500mV
M 500µs
CH1 3.00V

A.5. Red de pruebas en campus universitario de la PUCP

La red de pruebas se instaló en el campus de la Pontificia Universidad Católica del Perú, en los edificios más altos y/o ubicados estratégicamente. Esta red permite hacer varias pruebas sobre comunicaciones inalámbricas con la diferencia de que, si se compara con una red de laboratorio, permite tener un escenario de pruebas más parecido a la realidad.

La red está formada por nueve nodos. Cada nodo dispone de un mástil, una conexión de red con una *IP* asignada y un punto de conexión a la red eléctrica general. Mediante cada punto se pueden generar múltiples variantes de configuraciones de red para hacer pruebas tanto de configuración, como de software o hardware.

Configuración de la red de pruebas dentro del campus de la PUCP:

- Direcciones IP: De la 192.168.238.21 hasta la 192.168.238.29 (nueve puntos).
- VLan associada: Es la 238.
- Máscara de red: 255.255.255.0.
- Puerta de enlace: 192.168.238.10.


GTRed
LA RED LABORATORIO DEL GRUPO DE TELECOMUNICACIONES RURALES
Av. José de la Riva-Agüero
Av. Universitaria
CEPREPUC
Facultad de Ciencias Sociales
Biblioteca Central
Estudios Generales Letras
Enlace de Larga Distancia
(Más de 40 km.)
Puntos de Red
Edificios en Red
Puntos Conectados

A.6. Configuraciones e imágenes de las pruebas en el campus de la PUCP

A.6.1. Archivos de configuración de los equipos para las pruebas en el campus de la PUCP

Configuración Avila GW2348-4 (Pabellón V)

Archivo *network*:

```
#------------------------------
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#------------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.23'
option 'netmask' '255.255.255.0'
#------------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 10.10.10.2
option netmask 255.255.255.0
#------------------------------
config 'interface' 'wan1'
option ifname ath1
option proto static
option ipaddr 30.30.30.1
option netmask 255.255.255.0
#------------------------------
config 'interface' 'wan2'
```

```
option ifname ath2
option proto static
option ipaddr 50.50.50.1
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan3'
option ifname ath3
option proto static
option ipaddr 40.40.40.2
option netmask 255.255.255.0
#-----------------------------
```

Archivo *wireless*:

```
######################---######################
config wifi-device wifi0
option type atheros
option channel 2
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi0
option network wan0
option mode sta
option ssid op1
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi1
option type atheros
option channel 11
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi1
```

```
option network wan1
option mode ap
option ssid op3
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi2
option type atheros
option channel 3
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi2
option network wan2
option mode ap
option ssid op5
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi3
option type atheros
#option mode g  11g 802.11g  a 11a
#option channel 160
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi3
option network wan3
option mode sta
option ssid op4
#option encryption wep
#option hidden ...0...
#option key 1
```

```
#option key1 0123456789
######################---######################
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common

START=90

start() {

iwconfig ath0 txpower 10
iwconfig ath1 txpower 10
iwconfig ath2 txpower 10
iwconfig ath3 txpower 10

iwconfig ath0 rate 6M
iwconfig ath1 rate 6M
iwconfig ath2 rate 6M
iwconfig ath3 rate 6M

route add default gw 192.168.238.10

echo 0 > /proc/sys/dev/wifi0/diversity
echo 2 > /proc/sys/dev/wifi0/txantenna
echo 2 > /proc/sys/dev/wifi0/rxantenna

echo 0 > /proc/sys/dev/wifi1/diversity
echo 2 > /proc/sys/dev/wifi1/txantenna
echo 2 > /proc/sys/dev/wifi1/rxantenna

echo 0 > /proc/sys/dev/wifi2/diversity
echo 2 > /proc/sys/dev/wifi2/txantenna
echo 2 > /proc/sys/dev/wifi2/rxantenna

echo 0 > /proc/sys/dev/wifi3/diversity
echo 2 > /proc/sys/dev/wifi3/txantenna
echo 2 > /proc/sys/dev/wifi3/rxantenna
```

```
ntpdate -u -b 200.16.6.30

}
```

Archivo *root*:

```
# Syntax for lines is : minute hour day month dayofweek command #
01 * * * * /usr/local/bin/latencia1
01 * * * * /usr/local/bin/latencia2
56 * * * 5 /usr/local/bin/bw-iperf-destino4
57 * * * 1 /usr/local/bin/bw-iperf-origen5
56 * * * 2 /usr/local/bin/bw-iperf-destino1
57 * * * 3 /usr/local/bin/bw-iperf-origen3
```

Archivo *latencia1*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:01 MINS
#-----------------------------------------------------------------
# PING
#-----------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &

DATA=0
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0

TMPFILE01=/tmp/tmp.temp01
```

```
ping -q 10.10.10.1 -w 3250 > $TMPFILE01 #ip op1

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE01 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $8}')

#---------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#---------------------------------------------------------------

iwconfig ath0 > $TMPFILE01 #ath0 corresponde a op1

SL=$(cat $TMPFILE01 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE01 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP1.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *latencia2*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:01 MINS
#---------------------------------------------------------------
# PING
#---------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &
```

```
DATA=0
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0

TMPFILE02=/tmp/tmp.temp02

ping -q 40.40.40.1 -w 3250 > $TMPFILE02 #ip op4

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE02 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE02 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE02 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE02 |grep "min avg max" | awk '{print $8}')

#---------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#---------------------------------------------------------------

iwconfig ath3 > $TMPFILE02 #ath3 corresponde a op4

SL=$(cat $TMPFILE02 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE02 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP4.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen3*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 MINS
```

```
#-------------------------------------------------------------------
# BW (IPERF)
#-------------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 30.30.30.2 -i 5 -t 60 -p 5001 #ip origen OP3

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen5*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 MINS
#-------------------------------------------------------------------
# BW (IPERF)
#-------------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 50.50.50.2 -i 5 -t 60 -p 5001 #ip origen OP5

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino1*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-------------------------------------------------------------------
# BW (IPERF)
#-------------------------------------------------------------------
ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE12=/tmp/tmp.temp12
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE12 & #iperf -s en bg
```

```
sleep 150

B=$(cat $TMPFILE12 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP1-iperf_MARTES.txt
#Guarda el resultado del iperf -s

killall iperf
ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino4*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------
ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE11=/tmp/tmp.temp11
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE11 & #iperf -s en bg
sleep 150

B=$(cat $TMPFILE11 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP4-iperf_VIERNES.txt
#Guarda el resultado del iperf -s

killall iperf
ntpdate -u -b 200.16.6.30 &
```

Configuración Alix 2C0 (CEPREPUCP)

Archivo *network*:

```
#-----------------------------
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#-----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.25'
option 'netmask' '255.255.255.0'
#-----------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 50.50.50.2
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan1'
option ifname ath1
option proto static
option ipaddr 60.60.60.1
option netmask 255.255.255.0
#-----------------------------
```

Archivo *wireless*:

```
######################---######################
config wifi-device wifi0
option type atheros
option channel 3
#option distance ...100...
option disabled 0
```

```
#---
config wifi-iface
option device wifi0
option network wan0
option mode sta
option ssid op5
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi1
option type atheros
#option mode g  11g 802.11g  a 11a
#option channel 160
option channel 4
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi1
option network wan1
option mode ap
option ssid op6
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common

START=90

start() {

iwconfig ath0 txpower 23 #R52H
```

```
iwconfig ath1 txpower 23 #R52H

iwconfig ath0 rate 6M
iwconfig ath1 rate 6M

route add default gw 192.168.238.10

echo 0 > /proc/sys/dev/wifi0/diversity
echo 2 > /proc/sys/dev/wifi0/txantenna
echo 2 > /proc/sys/dev/wifi0/rxantenna

echo 0 > /proc/sys/dev/wifi1/diversity
echo 2 > /proc/sys/dev/wifi1/txantenna
echo 2 > /proc/sys/dev/wifi1/rxantenna

ntpdate -u -b 200.16.6.30

}
```

Archivo *root*:

```
# Syntax for lines is : minute hour day month dayofweek command #
01 * * * * /usr/local/bin/latencia
56 * * * 1 /usr/local/bin/bw-iperf-destino5
57 * * * 2 /usr/local/bin/bw-iperf-origen6
```

Archivo *latencia*:

```
#!/bin/sh

#-----------------------------------------------------------------
# PING
#-----------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &
```

```
DATA=0
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0

TMPFILE01=/tmp/tmp.temp01

ping -q 50.50.50.1 -w 3250 > $TMPFILE01 #ip op5

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE01 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $8}')

#-----------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#-----------------------------------------------------------------

iwconfig ath0 > $TMPFILE01 #ath0 corresponde a op5

SL=$(cat $TMPFILE01 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE01 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP5.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen6*:

```
#!/bin/sh
```

```
# Sincronizar el cron para que arranque a las *:57 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 60.60.60.2 -i 5 -t 60 -p 5001 #ip origen OP6

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino5*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE11=/tmp/tmp.temp11
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE11 & #iperf -s en bg
sleep 150

B=$(cat $TMPFILE11 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP5-iperf_LUNES.txt
#Guarda el resultado del iperf -s

killall iperf

ntpdate -u -b 200.16.6.30 &
```

Configuración Alix 2C0 (Pabellón B)

Archivo *network*:

```
#-----------------------------
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#-----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.24'
option 'netmask' '255.255.255.0'
#-----------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 70.70.70.2
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan1'
option ifname ath1
option proto static
option ipaddr 40.40.40.1
option netmask 255.255.255.0
#-----------------------------
```

Archivo *wireless*:

```
######################---######################
config wifi-device wifi0
option type atheros
option channel 1
#option distance ...100...
option disabled 0
```

```
#---
config wifi-iface
option device wifi0
option network wan0
option mode sta
option ssid op7
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi1
option type atheros
option channel 10
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi1
option network wan1
option mode ap
option ssid op4
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common

START=90

start() {

iwconfig ath0 txpower 16 #SR2
iwconfig ath1 txpower 16 #SR2
```

```
iwconfig ath0 rate 6M
iwconfig ath1 rate 6M

route add default gw 192.168.238.10

echo 0 > /proc/sys/dev/wifi0/diversity
echo 2 > /proc/sys/dev/wifi0/txantenna
echo 2 > /proc/sys/dev/wifi0/rxantenna

echo 0 > /proc/sys/dev/wifi1/diversity
echo 2 > /proc/sys/dev/wifi1/txantenna
echo 2 > /proc/sys/dev/wifi1/rxantenna

ntpdate -u -b 200.16.6.30

}
```

Archivo *root*:

```
# Syntax for lines is : minute hour day month dayofweek command #
01 * * * * /usr/local/bin/latencia
56 * * * 4 /usr/local/bin/bw-iperf-destino7
57 * * * 5 /usr/local/bin/bw-iperf-origen4
```

Archivo *latencia*:

```
#!/bin/sh

#-----------------------------------------------------------------
# PING
#-----------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &

DATA=0
```

```
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0

TMPFILE01=/tmp/tmp.temp01

ping -q 70.70.70.1 -w 3250 > $TMPFILE01 #ip op7

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE01 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $8}')

#-----------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#-----------------------------------------------------------------

iwconfig ath0 > $TMPFILE01 #ath0 corresponde a op7

SL=$(cat $TMPFILE01 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE01 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP7.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen4*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 MINS
#-----------------------------------------------------------------
```

```
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 40.40.40.2 -i 5 -t 60 -p 5001 #ip origen OP4

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino7*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE11=/tmp/tmp.temp11
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE11 & #iperf -s en bg
sleep 150

B=$(cat $TMPFILE11 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP7-iperf_JUEVES.txt
#Guarda el resultado del iperf -s

killall iperf

ntpdate -u -b 200.16.6.30 &
```

Configuración Avila GW2348-4 (McGregor)

Archivo *network*:

```
#-----------------------------
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#-----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.22'
option 'netmask' '255.255.255.0'
#-----------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 20.20.20.1
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan1'
option ifname ath1
option proto static
option ipaddr 60.60.60.2
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan2'
option ifname ath2
option proto static
option ipaddr 30.30.30.2
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan3'
option ifname ath3
option proto static
```

```
option ipaddr 70.70.70.1
option netmask 255.255.255.0
#-----------------------------
```

Archivo *wireless*:

```
######################---######################
config wifi-device wifi0
option type atheros
option channel 11
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi0
option network wan0
option mode ap
option ssid op2
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi1
option type atheros
option channel 4
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi1
option network wan1
option mode sta
option ssid op6
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
```

```
config wifi-device wifi2
option type atheros
option channel 11
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi2
option network wan2
option mode sta
option ssid op3
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
########################---#######################
config wifi-device wifi3
option type atheros
option channel 1
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi3
option network wan3
option mode ap
option ssid op7
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
########################---#######################
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common

START=90
```

```
start() {

iwconfig ath0 txpower 16 #XR2
iwconfig ath1 txpower 16 #XR2
iwconfig ath2 txpower 16 #XR2
iwconfig ath3 txpower 16 #XR2

iwconfig ath0 rate 6M
iwconfig ath1 rate 6M
iwconfig ath2 rate 6M
iwconfig ath3 rate 6M

route add default gw 192.168.238.10

echo 0 > /proc/sys/dev/wifi0/diversity
echo 2 > /proc/sys/dev/wifi0/txantenna
echo 2 > /proc/sys/dev/wifi0/rxantenna

echo 0 > /proc/sys/dev/wifi1/diversity
echo 2 > /proc/sys/dev/wifi1/txantenna
echo 2 > /proc/sys/dev/wifi1/rxantenna

echo 0 > /proc/sys/dev/wifi2/diversity
echo 2 > /proc/sys/dev/wifi2/txantenna
echo 2 > /proc/sys/dev/wifi2/rxantenna

echo 0 > /proc/sys/dev/wifi3/diversity
echo 2 > /proc/sys/dev/wifi3/txantenna
echo 2 > /proc/sys/dev/wifi3/rxantenna

ntpdate -u -b 200.16.6.30

}
```

Archivo *root*:

```
# Syntax for lines is : minute hour day month dayofweek command #
01 * * * * /usr/local/bin/latencia1
01 * * * * /usr/local/bin/latencia2
```

```
56 * * * 2 /usr/local/bin/bw-iperf-destino6
57 * * * 4 /usr/local/bin/bw-iperf-origen7
56 * * * 3 /usr/local/bin/bw-iperf-destino3
57 * * * 1 /usr/local/bin/bw-iperf-origen2
```

Archivo *latencia1*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:01 MINS
#-----------------------------------------------------------------
# PING
#-----------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &

DATA=0
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0

TMPFILE01=/tmp/tmp.temp01

ping -q 30.30.30.1 -w 3250 > $TMPFILE01 #ip op3

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE01 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $8}')
```

```
#-------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#-------------------------------------------------------------

iwconfig ath2 > $TMPFILE01 #ath2 corresponde a op3

SL=$(cat $TMPFILE01 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE01 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP3.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *latencia2*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:01 MINS
#-------------------------------------------------------------
# PING
#-------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &

DATA=0
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0
```

```
TMPFILE02=/tmp/tmp.temp02

ping -q 60.60.60.1 -w 3250 > $TMPFILE02 #ip op6

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE02 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE02 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE02 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE02 |grep "min avg max" | awk '{print $8}')

#-------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#-------------------------------------------------------------

iwconfig ath1 > $TMPFILE02 #ath1 corresponde a op6

SL=$(cat $TMPFILE02 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE02 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP6.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen2*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 MINS
#-------------------------------------------------------------
# BW (IPERF)
#-------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 20.20.20.2 -i 5 -t 60 -p 5001 #ip origen OP2
```

```
ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen7*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 70.70.70.2 -i 5 -t 60 -p 5001 #ip origen OP7

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino3*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE12=/tmp/tmp.temp12
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE12 & #iperf -s en bg
sleep 150

B=$(cat $TMPFILE12 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP3-iperf_MIERCOLES.txt
#Guarda el resultado del iperf -s
```

```
killall iperf

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino6*:

```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-----------------------------------------------------------------
# BW (IPERF)
#-----------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE11=/tmp/tmp.temp11
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE11 & #iperf -s en bg
sleep 150

B=$(cat $TMPFILE11 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP6-iperf_MARTES.txt
#Guarda el resultado del iperf -s

killall iperf

ntpdate -u -b 200.16.6.30 &
```

Configuración Pronghorn SBC 250 (Biblioteca)

Archivo *network*:

```
#-----------------------------
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#-----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.21'
option 'netmask' '255.255.255.0'
#-----------------------------
config 'interface' 'wan0'
option ifname ath0
option proto static
option ipaddr 10.10.10.1
option netmask 255.255.255.0
#-----------------------------
config 'interface' 'wan1'
option ifname ath1
option proto static
option ipaddr 20.20.20.2
option netmask 255.255.255.0
#-----------------------------
```

Archivo *wireless*:

```
######################---######################
config wifi-device wifi0
option type atheros
option channel 2
#option distance ...100...
option disabled 0
```

```
#---
config wifi-iface
option device wifi0
option network wan0
option mode ap
option ssid op1
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
config wifi-device wifi1
option type atheros
option channel 11
#option distance ...100...
option disabled 0
#---
config wifi-iface
option device wifi1
option network wan1
option mode sta
option ssid op2
#option encryption wep
#option hidden ...0...
#option key 1
#option key1 0123456789
######################---######################
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common

START=90

start() {

iwconfig ath0 txpower 16 #SR2
iwconfig ath1 txpower 16 #SR2
```

```
iwconfig ath0 rate 6M
iwconfig ath1 rate 6M

route add default gw 192.168.238.10

echo 0 > /proc/sys/dev/wifi0/diversity
echo 2 > /proc/sys/dev/wifi0/txantenna
echo 2 > /proc/sys/dev/wifi0/rxantenna

echo 0 > /proc/sys/dev/wifi1/diversity
echo 2 > /proc/sys/dev/wifi1/txantenna
echo 2 > /proc/sys/dev/wifi1/rxantenna

ntpdate -u -b 200.16.6.30

}
```

Archivo *root*:

```
# Syntax for lines is : minute hour day month dayofweek command #
01 * * * * /usr/local/bin/latencia
56 * * * 1 /usr/local/bin/bw-iperf-destino2
57 * * * 2 /usr/local/bin/bw-iperf-origen1
```

Archivo *latencia*:

```
#!/bin/sh

#-----------------------------------------------------------------
# PING
#-----------------------------------------------------------------

#Se inicializan las variables a 0
#Se crea un archivo temporal donde se guardaran los datos
# que despues se transferiran al archivo /root/ping-result...
#Al final se hace un echo para alduntar una linia mas al archivo

ntpdate -u -b 200.16.6.30 &

DATA=0
```

```
PLOSS=0
MIN=0
AVG=0
MAX=0

SL=0
NL=0

TMPFILE01=/tmp/tmp.temp01

ping -q 20.20.20.1 -w 3250 > $TMPFILE01 #ip op2

DATA=$(date '+%F %H:%M')
PLOSS=$(cat $TMPFILE01 | grep "packet loss" | awk '{print $7}')
MIN=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $6}')
AVG=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $7}')
MAX=$(sed  -e  's/\// /g' $TMPFILE01 |grep "min avg max" | awk '{print $8}')

#-----------------------------------------------------------------
# NIVEL DE SEÑAL Y NIVEL DE RUIDO
#-----------------------------------------------------------------

iwconfig ath1 > $TMPFILE01 #ath1 corresponde a op2

SL=$(cat $TMPFILE01 |grep "Signal level" | awk '{print $3 $4}' | cut -d"="-f2)
NL=$(cat $TMPFILE01 |grep "Noise level" | awk '{print $6 $7}' | cut -d"="-f2)



#DATA PLOSS MIN AVG MAX Signal level Noise level
echo "$DATA  $PLOSS  $MIN  $AVG  $MAX  $SL  $NL" >> /root/result-OP2.txt

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-origen1*:

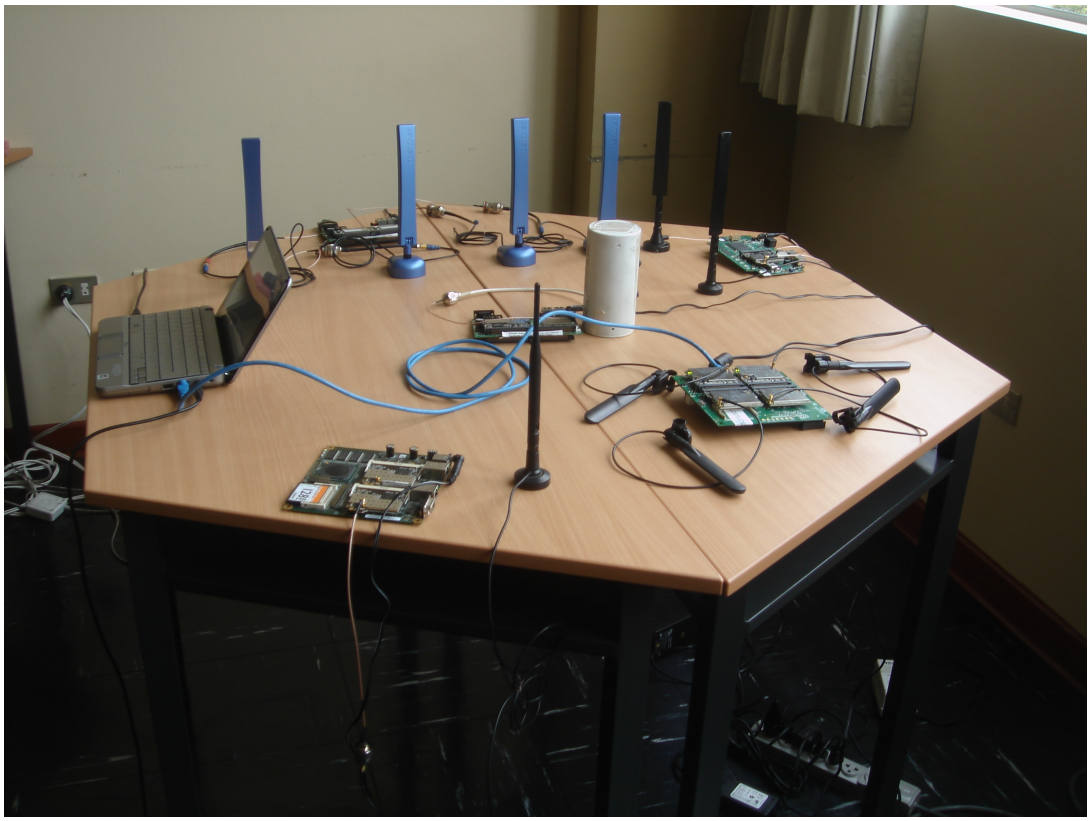
```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:57 MINS
#-----------------------------------------------------------------
```

```
# BW (IPERF)
#-------------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

iperf -c 10.10.10.2 -i 5 -t 60 -p 5001 #ip origen OP1

ntpdate -u -b 200.16.6.30 &
```

Archivo *bw-iperf-destino2*:

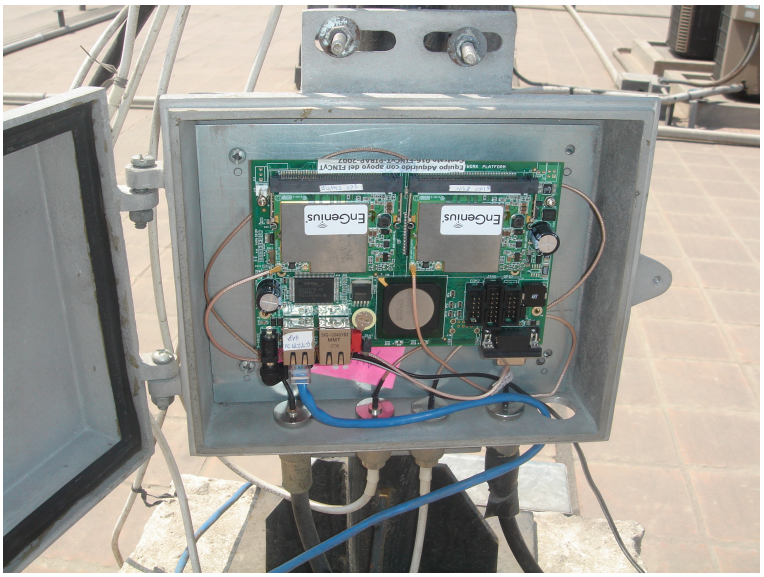
```
#!/bin/sh
# Sincronizar el cron para que arranque a las *:56 MINS
#-------------------------------------------------------------------
# BW (IPERF)
#-------------------------------------------------------------------

ntpdate -u -b 200.16.6.30 &

B=0
DATAIPERF=0

TMPFILE11=/tmp/tmp.temp11
DATAIPERF=$(date '+%F %H:%M')

iperf -p 5001 -s > $TMPFILE11 & #iperf -s en bg
sleep 150

B=$(cat $TMPFILE11 | grep "/sec" | awk '{print $7 $8}')

# BW - Iperf (Bandwidth_Xbits/sec)
echo "$DATAIPERF $B" >> /root/result-OP2-iperf_LUNES.txt
#Guarda el resultado del iperf -s

killall iperf

ntpdate -u -b 200.16.6.30 &
```

A.6.2. Fotos del montaje en el laboratorio de los equipos para las pruebas en el campus de la PUCP



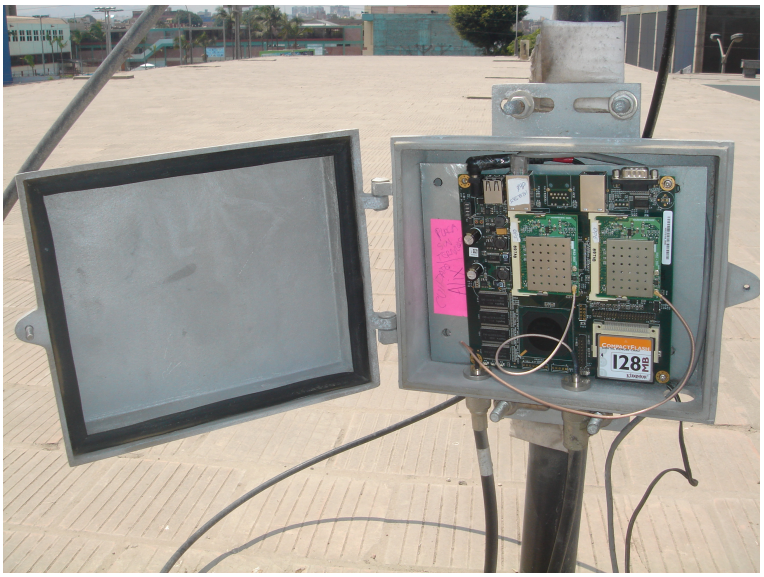
A.6.3. Fotos de las pruebas el campus de la PUCP

Fotos de los equipos situados en la azotea del edificio V

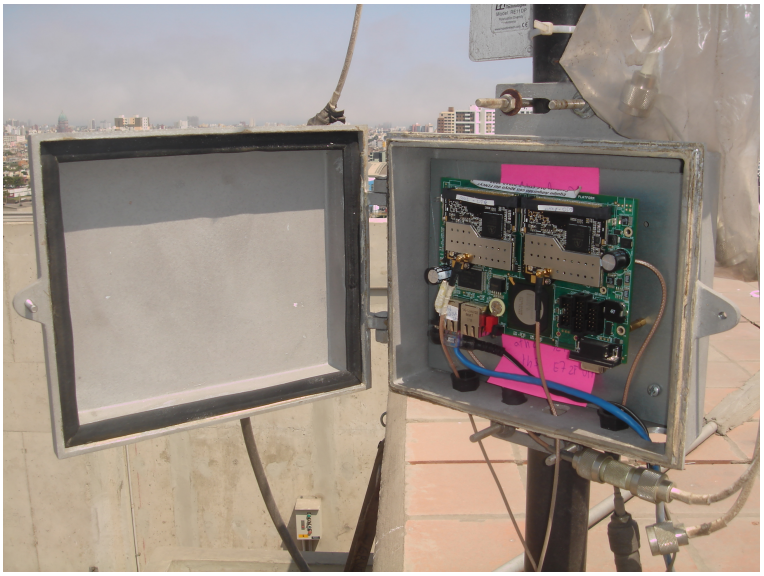

EnGenius
EnGenius

Fotos de los equipos situados en la azotea del edificio B

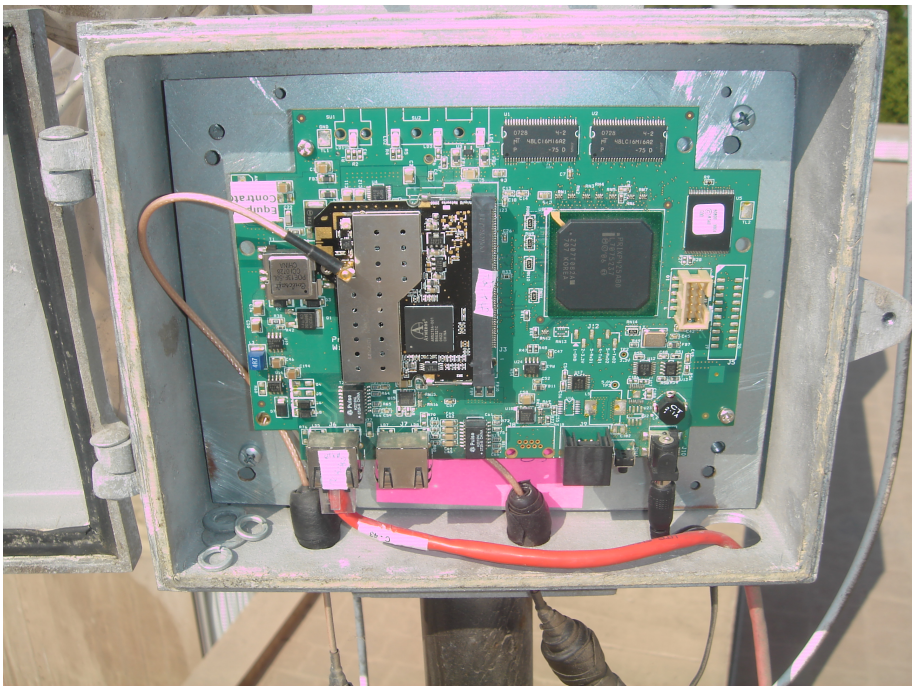
Fotos de los equipos situados en la azotea del edificio CEPREPUCP

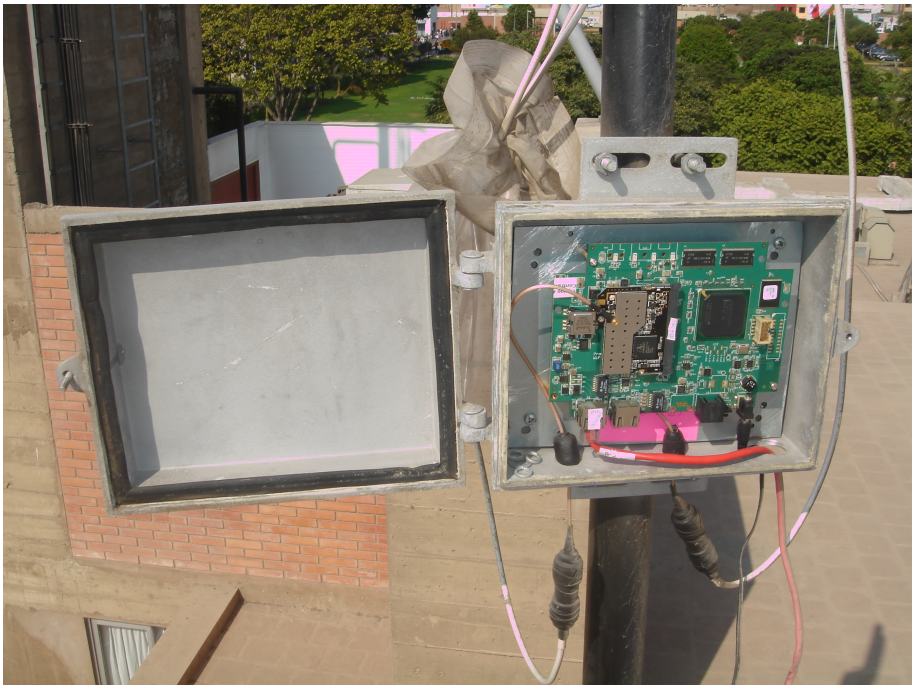

128 MB

Fotos de los equipos situados en la azotea del edificio McGregor

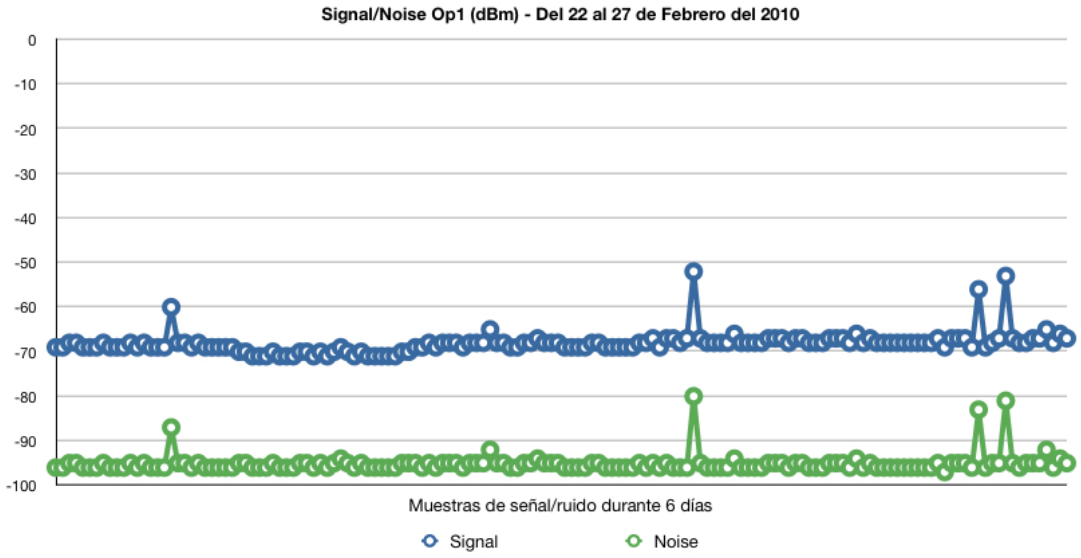


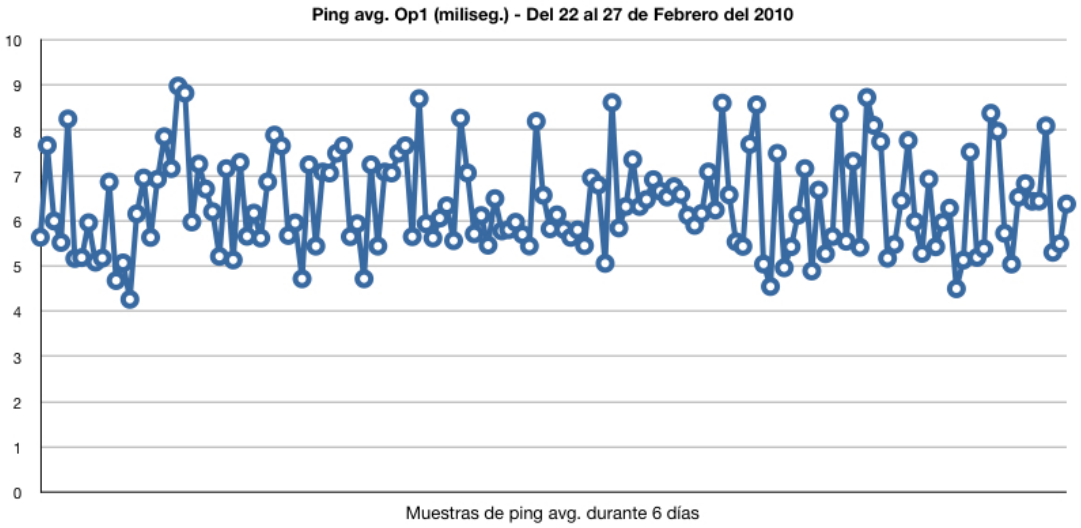
Fotos de los equipos situados en la azotea del edificio de la biblioteca central

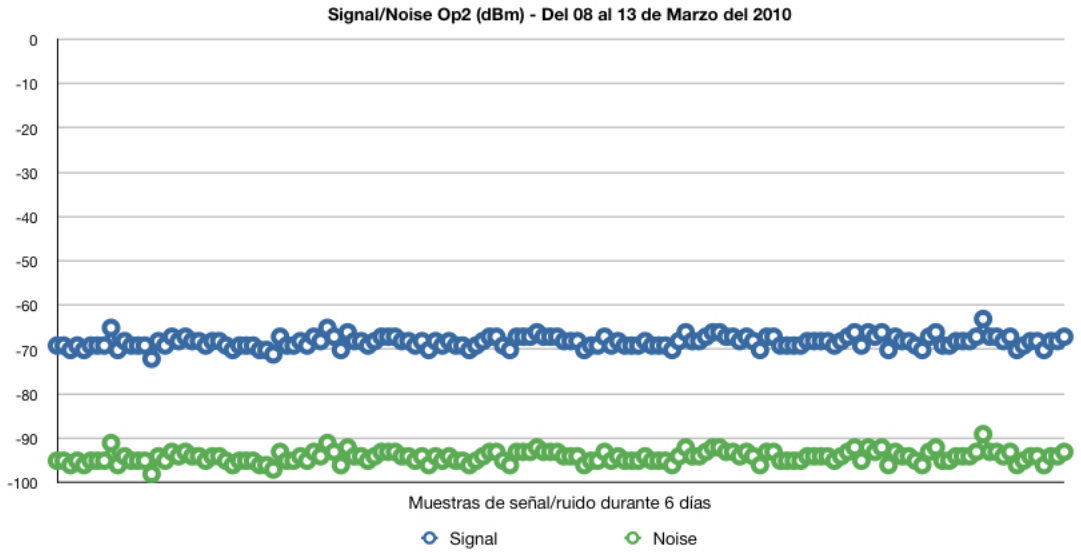


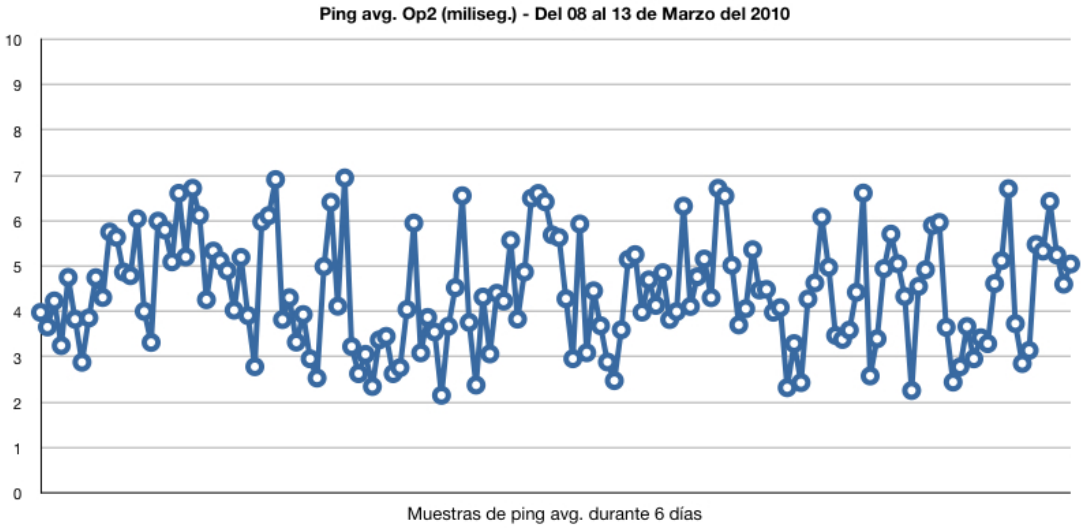


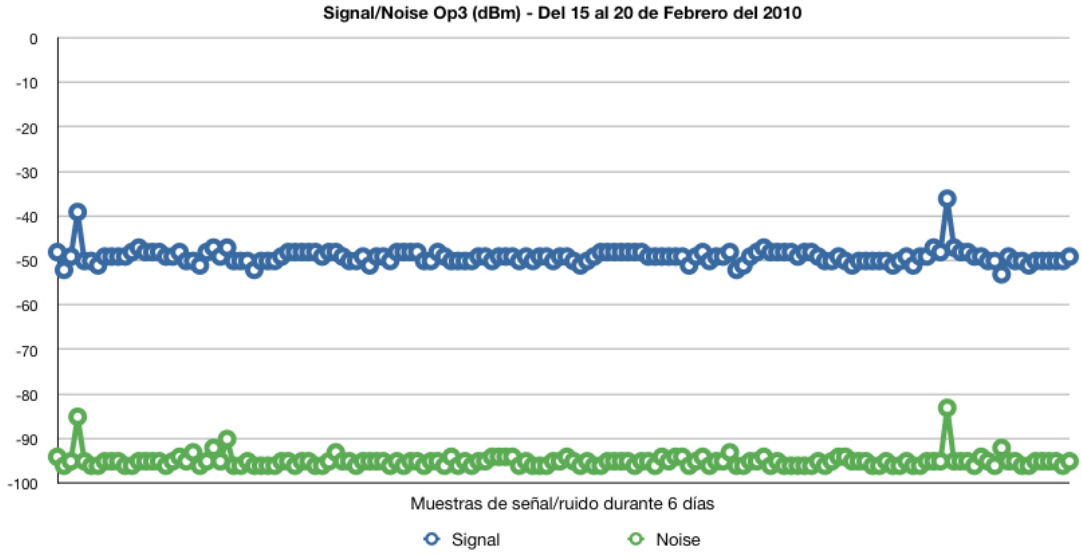
A.7. Resultados semanales de las pruebas en el campus de la PUCP

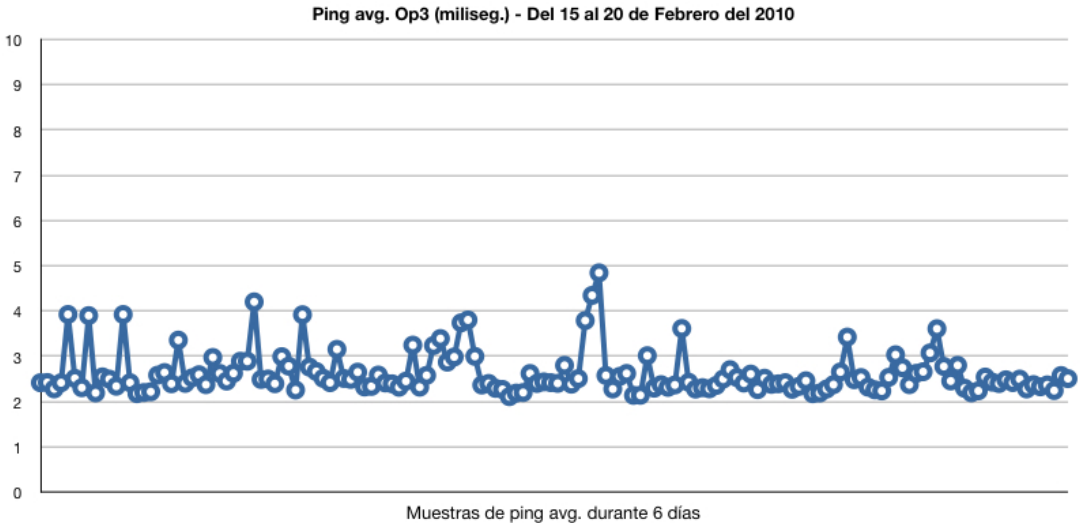


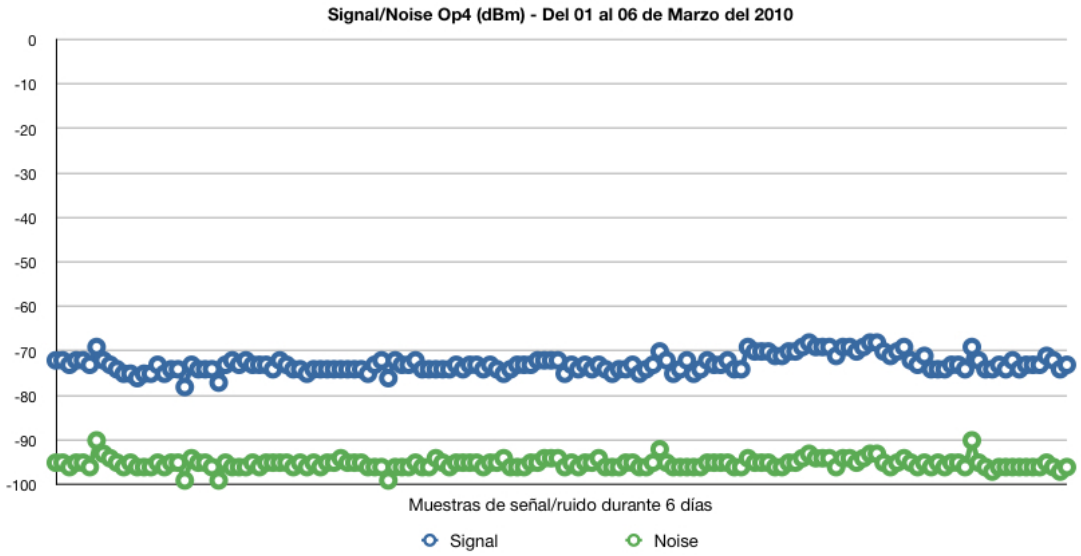


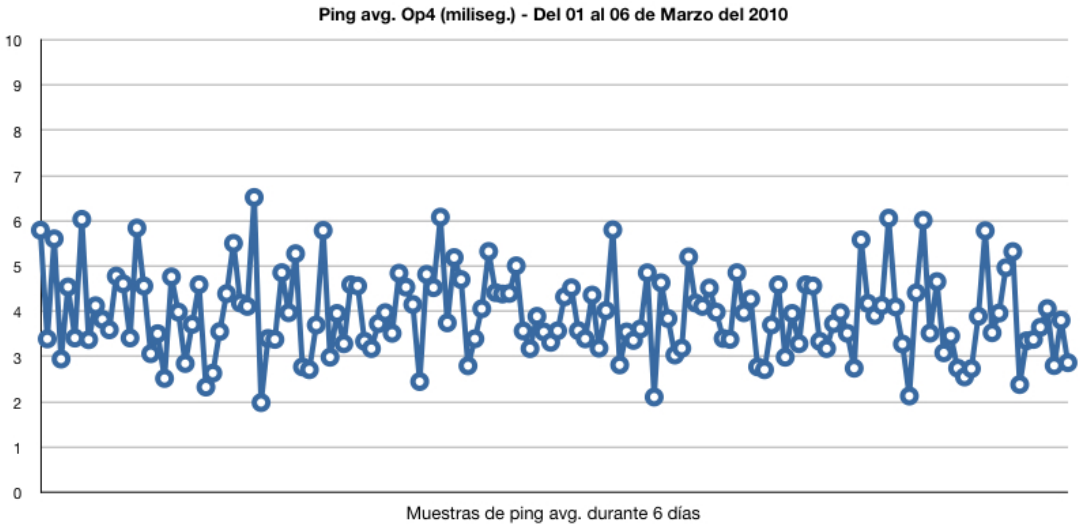

Signal/Noise Op2 (dBm) - Del 08 al 13 de Marzo del 2010
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Muestras de señal/ruido durante 6 días
Signal
Noise

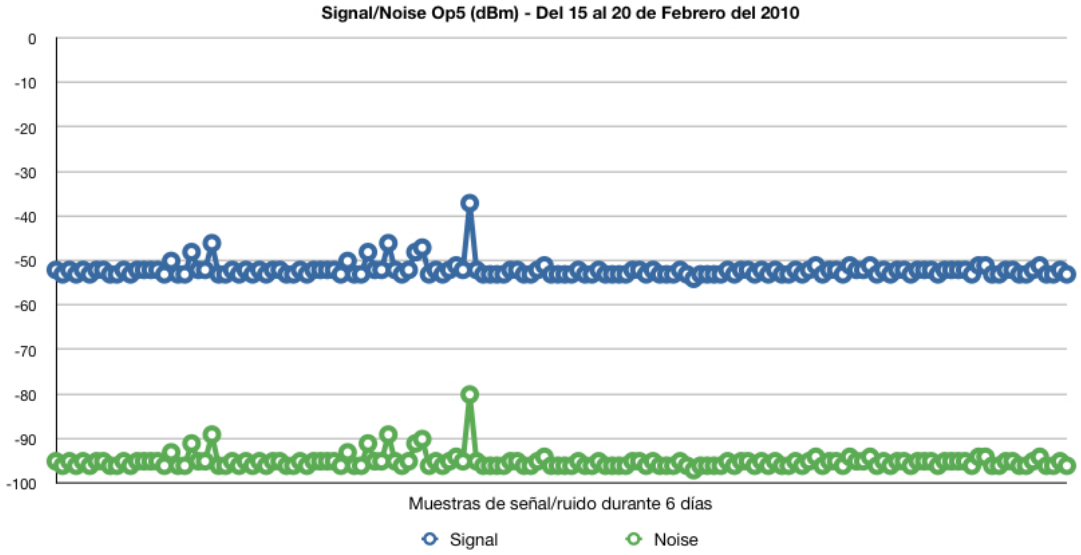

Ping avg. Op2 (miliseg.) - Del 08 al 13 de Marzo del 2010
10
9
8
7
6
5
4
3
2
1
0
Muestras de ping avg. durante 6 días

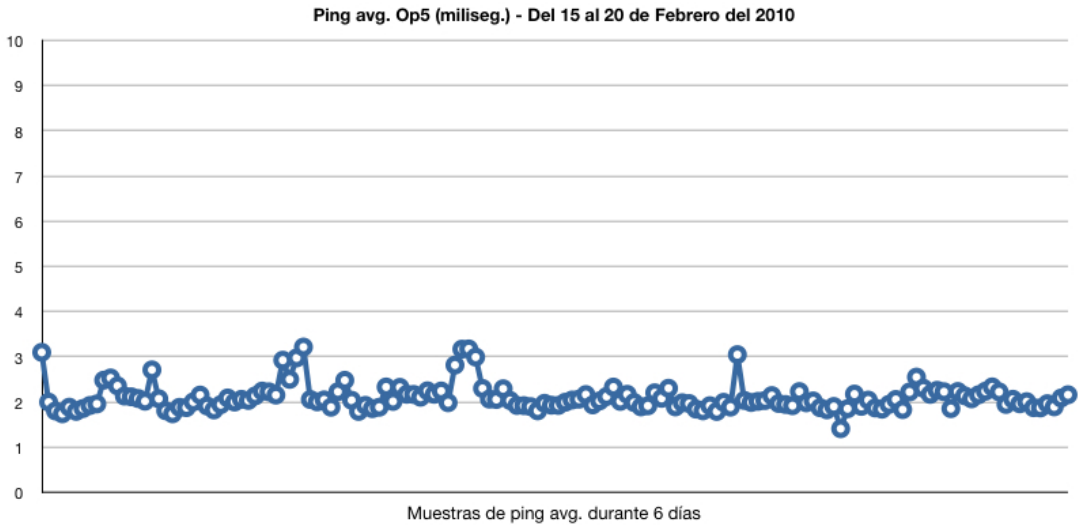

Signal/Noise Op3 (dBm) - Del 15 al 20 de Febrero del 2010
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Muestras de señal/ruido durante 6 días
Signal
Noise

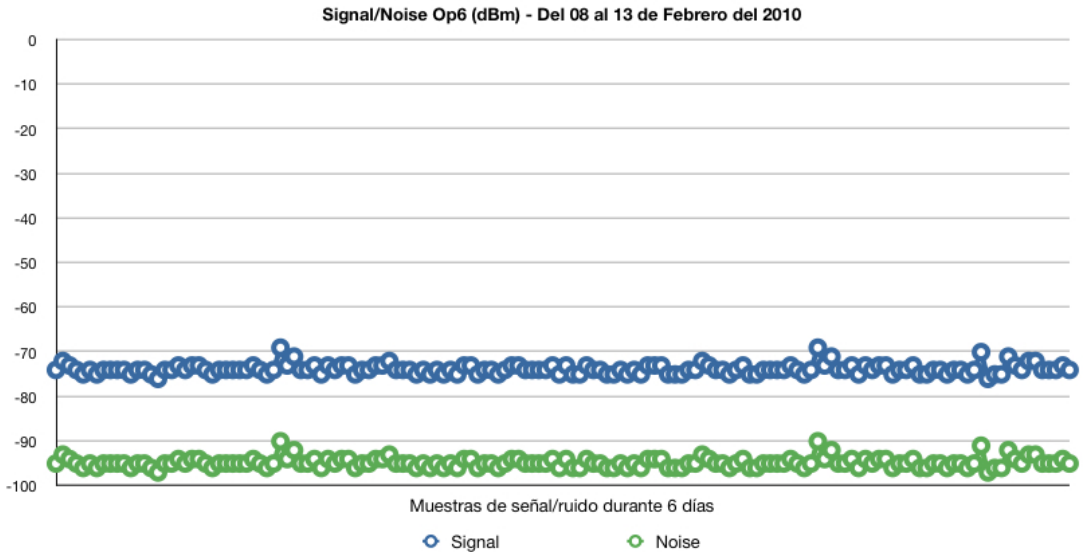

Ping avg. Op3 (miliseg.) - Del 15 al 20 de Febrero del 2010
10
9
8
7
6
5
4
3
2
1
0
Muestras de ping avg. durante 6 días

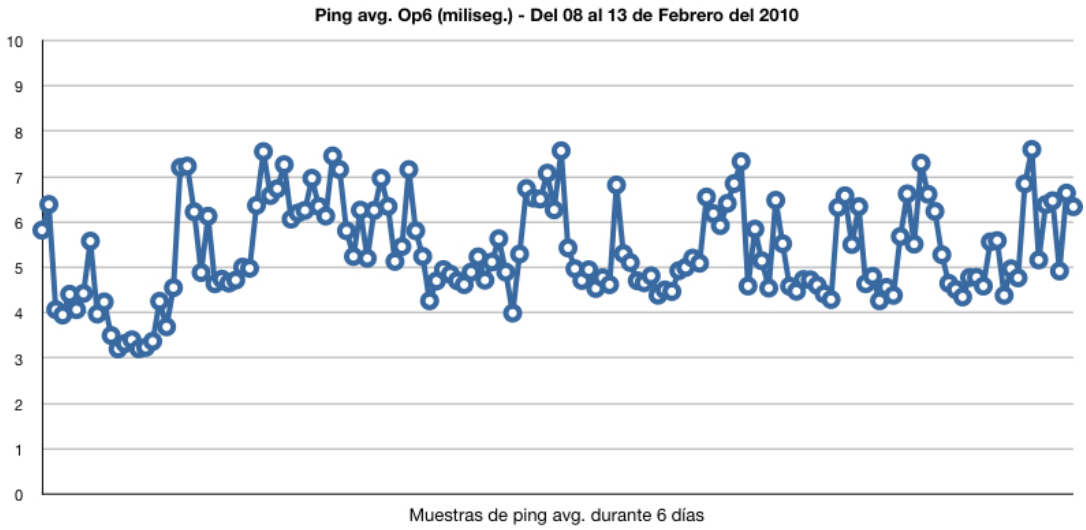

Signal/Noise Op4 (dBm) - Del 01 al 06 de Marzo del 2010
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Muestras de señal/ruido durante 6 días
Signal
Noise

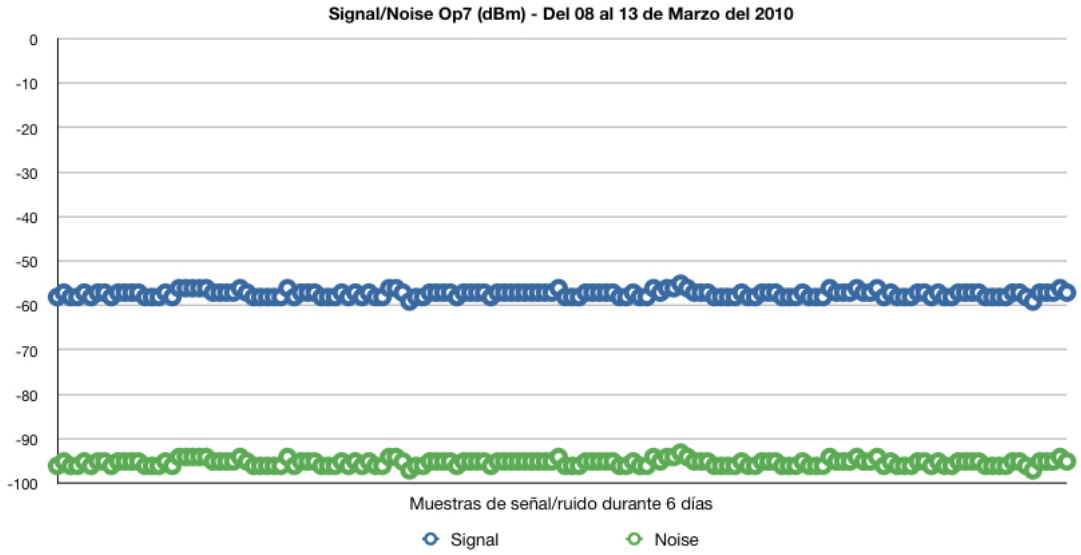

Ping avg. Op4 (miliseg.) - Del 01 al 06 de Marzo del 2010
10
9
8
7
6
5
4
3
2
1
0
Muestras de ping avg. durante 6 días

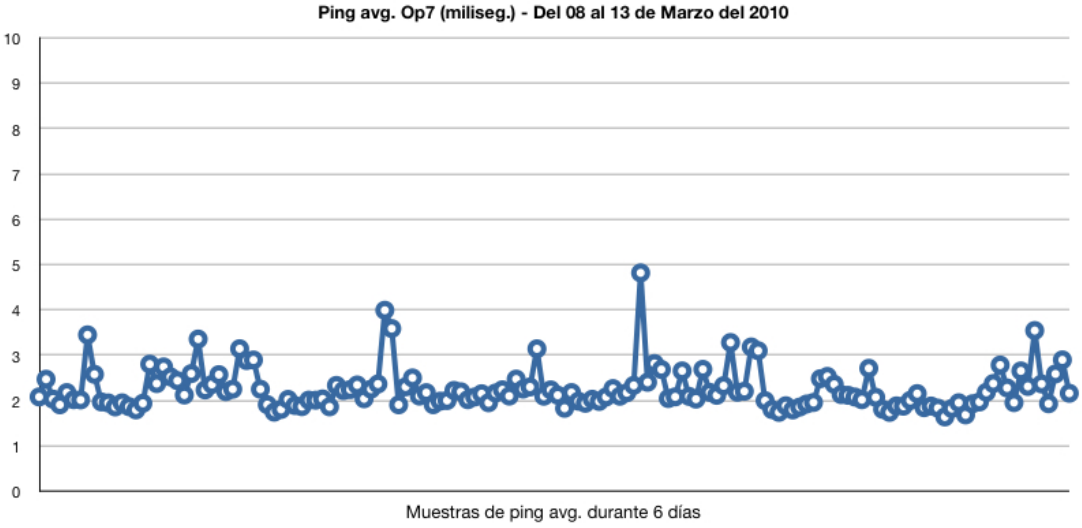

Signal/Noise Op5 (dBm) - Del 15 al 20 de Febrero del 2010
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Muestras de señal/ruido durante 6 días
Signal
Noise


Ping avg. Op5 (miliseg.) - Del 15 al 20 de Febrero del 2010
10
9
8
7
6
5
4
3
2
1
0
Muestras de ping avg. durante 6 días


Signal/Noise Op6 (dBm) - Del 08 al 13 de Febrero del 2010
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Muestras de señal/ruido durante 6 días
Signal
Noise


Ping avg. Op6 (miliseg.) - Del 08 al 13 de Febrero del 2010
10
9
8
7
6
5
4
3
2
1
0
Muestras de ping avg. durante 6 días


Signal/Noise Op7 (dBm) - Del 08 al 13 de Marzo del 2010
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Muestras de señal/ruido durante 6 días
Signal
Noise


Ping avg. Op7 (miliseg.) - Del 08 al 13 de Marzo del 2010
10
9
8
7
6
5
4
3
2
1
0
Muestras de ping avg. durante 6 días

A.8. Configuración e imágenes de la prueba en campo

A.8.1. Archivos de configuración

Configuración AP

Archivo *network*:

```
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#-----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.21'
option 'netmask' '255.255.255.0'
#-----------------------------
config 'interface' 'wan'
option ifname ath0
option proto static
option ipaddr 10.10.10.1
option netmask 255.255.255.0
```

Archivo *wireless*:

```
config wifi-device wifi0
option type atheros
option channel 5
option distance 30000
#
config wifi-iface
option device wifi0
option network wan
option mode ap
option ssid openwrt
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common

START=90

start() {

echo 0 > /proc/sys/dev/wifi0/diversity
echo 1 > /proc/sys/dev/wifi0/txantenna
echo 1 > /proc/sys/dev/wifi0/rxantenna

}
```

Configuración AP

Archivo *network*:

```
config 'interface' 'loopback'
option 'ifname' 'lo'
option 'proto' 'static'
option 'ipaddr' '127.0.0.1'
option 'netmask' '255.0.0.0'
#-----------------------------
config 'interface' 'lan'
option 'ifname' 'eth0'
option 'type' 'bridge'
option 'proto' 'static'
option 'ipaddr' '192.168.238.25'
option 'netmask' '255.255.255.0'
#-----------------------------
config 'interface' 'wan'
option ifname ath0
option proto static
option ipaddr 10.10.10.2
option netmask 255.255.255.0
```

Archivo *wireless*:

```
config wifi-device wifi0
option type atheros
option channel 5
option distance 30000
#
config wifi-iface
option device wifi0
option network wan
option mode sta
option ssid openwrt
```

Archivo *redes*:

```
#!/bin/sh /etc/rc.common
```

```
START=90

start() {

echo 0 > /proc/sys/dev/wifi0/diversity
echo 1 > /proc/sys/dev/wifi0/txantenna
echo 1 > /proc/sys/dev/wifi0/rxantenna

}
```

A.8.2. Fotos de la prueba en campo en el río Napo

Bibliografía

[1] Francisco Javier Simó Reigadas. *Modelado y optimización de IEEE 802.11 para su aplicación en el despliegue de redes extensas en zonas rurales. España*. 2007.

[2] Matthew S. Gast. *802.11 wireless networks: the definitive guide*. Sebastopol, CA: O'Reilly, 2005.

[3] Josep Oriol Madriles Soriano. Diseño de una red de telecomunicación para la interconexión de datos y telefonía para municipios del departamento de cuzco. Technical report, 2008.

[4] Sandra Paulina Espinoza Avalos. Estudio de viabilidad técnica y económica para la migración de una red wifi a wimax en entornos rurales. Technical report, 2010.

[5] Matthew S. Gast. *Redes wireless 802.11*. Madrid: Anaya Multimedia, 2006.

[6] *GNU General Public License*. 3 edition, 2007.

[7] Luis Camacho, River Quispe, César Córdova, Leopoldo Liñán, and David Chávez. *WiFi Based Long Distance*. Pontifia Universidad Católica Del Perú, 2009.

[8] Mark G. Sobell. *A practical guide to Linux commands, editors, and shell programming*. Pearson Education, 2005.

[9] Arnold Robbins. *Linux programming by example*. Prentice Hall PTR, 2004.

[10] *MTC. Resolución Ministerial 777. Perú*. 2005.